

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平11-224184

(43)公開日 平成11年(1999) 8月17日

(51)Int.Cl.⁶

識別記号

F I

G 0 6 F 9/06
12/00
17/30

5 3 0
5 1 3

G 0 6 F 9/06 5 3 0 V
12/00 5 1 3 D
15/40 3 8 0 E

審査請求 有 請求項の数9 O L (全 24 頁)

(21)出願番号

特願平10-24459

(22)出願日

平成10年(1998) 2月 5 日

(71)出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72)発明者 北野 拓哉

東京都港区芝五丁目7番1号 日本電気株
式会社内

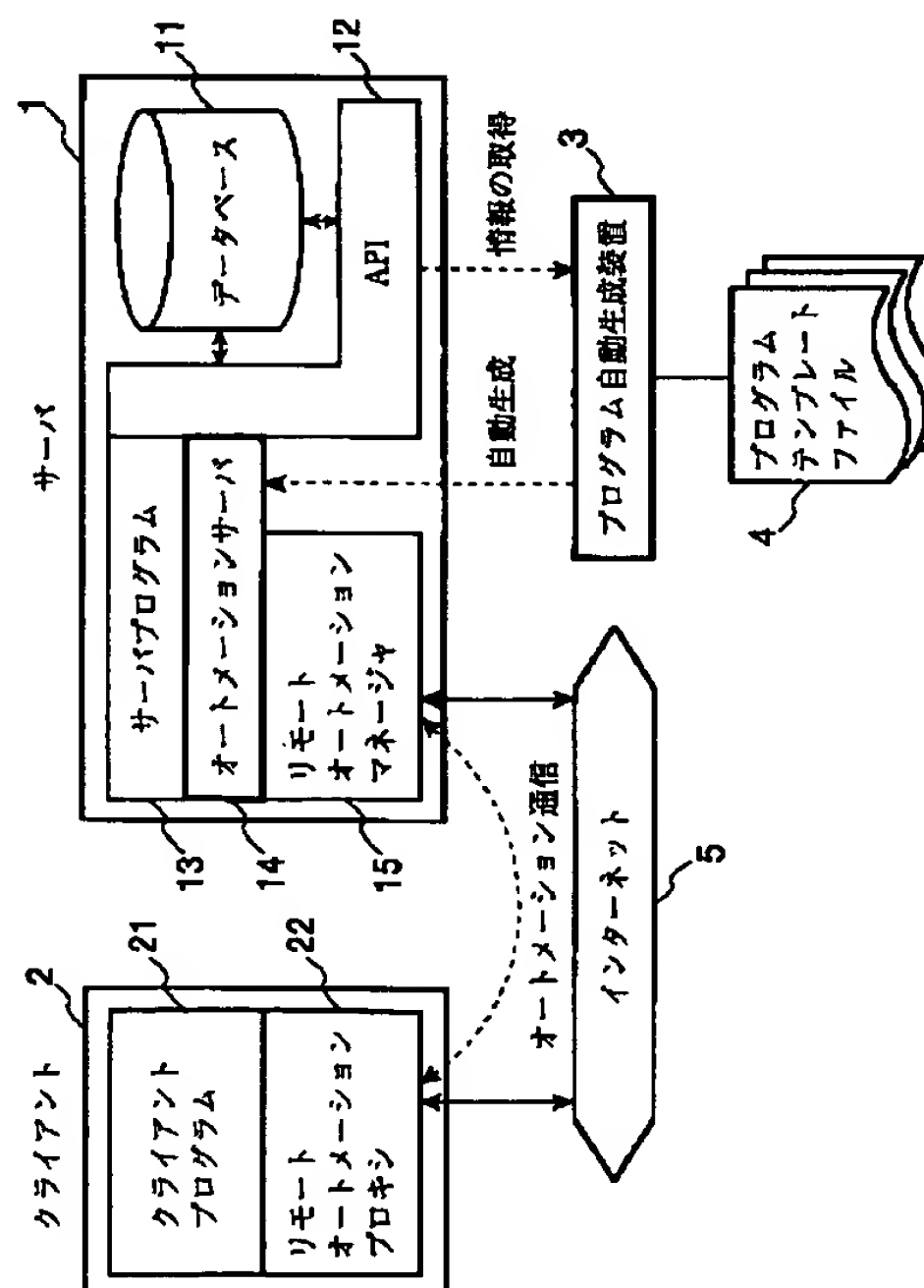
(74)代理人 弁理士 古澤 聡 (外1名)

(54)【発明の名称】 オブジェクト指向データベース操作プログラム生成システム及び方法

(57)【要約】

【課題】 既存のオブジェクト指向データベースを、既存のオートメーション通信の機能を利用して操作するプログラムを自動的に生成する。また、プログラムの開発・保守を容易にする。

【解決手段】 プログラム自動生成装置3は、データベース11からAPI12を介してそのデータベース11に定義されているスキーマ情報を取得し、取得したスキーマ情報とプログラムテンプレートファイル4を基にして、オートメーション通信のためのオートメーションサーバ14のためのプログラムを自動生成する。オートメーションサーバ14は、データベース11のスキーマを構成するクラスとその属性、メソッドを、オートメーション通信を利用するクライアントプログラム21に公開する。



【特許請求の範囲】

【請求項 1】オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得手段と、

前記スキーマを構成する各クラスに含まれるオブジェクトの操作に用いられるプログラムを生成するための第 1 のテンプレートを記憶する第 1 のテンプレート記憶手段と、

前記第 1 のテンプレート記憶手段から前記第 1 のテンプレートを読み出し、該読み出した第 1 のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するための第 1 のプログラムを生成するプログラム生成手段と、

前記プログラム生成手段が生成する前記スキーマを構成するクラス毎のプログラムが外部に公開すべきインタフェースを定義するためのインタフェース定義の生成に用いられる第 2 のテンプレートを記憶する第 2 のテンプレート記憶手段と、

前記第 2 のテンプレート記憶手段から第 2 のテンプレートを読み出し、該読み出した第 2 のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成手段と、を備えることを特徴とするオブジェクト指向データベース操作プログラム生成システム。

【請求項 2】前記プログラム生成手段が生成した前記第 1 のプログラムと前記インタフェース定義生成手段が生成した前記インタフェース定義とをリンクすることによって、実行可能なモジュールを作り上げるリンク手段と、をさらに備えることを特徴とする請求項 1 に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 3】前記スキーマを構成する各クラスに含まれるオブジェクトを外部から操作するための外部操作情報を生成するための第 3 のテンプレートを記憶する第 3 のテンプレート記憶手段と、

前記第 3 のテンプレート記憶手段から第 3 のテンプレートを読み出し、該読み出した第 3 のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成手段と、をさらに備えることを特徴とする請求項 1 または 2 に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 4】設定された情報によってインタフェースを介して外部から他のプログラムを操作するための第 2 の

プログラムを実行させるプログラム実行手段と、

前記外部操作情報生成手段が生成した外部操作情報を前記第 2 のプログラムに設定することによって、前記第 2 のプログラムが前記インタフェース定義手段によって定義されたインタフェース定義に対応するインタフェースを介して前記プログラム生成手段によって生成された前記第 1 のプログラムを操作することが可能となるように設定する設定手段と、を備えることを特徴とする請求項 3 に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 5】前記プログラム生成手段は、前記スキーマを構成する各クラス毎に、プロパティの取得関数を生成する手段と、プロパティの設定関数を生成する手段と、メソッドを生成する手段とを備えることを特徴とする請求項 1 乃至 4 のいずれか 1 項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 6】前記インタフェース定義によって定義されるインタフェースは、オートメーション通信を行うためのインタフェースであり、

前記外部操作情報は、オートメーション通信の機能を利用して前記プログラム生成手段が生成した前記第 1 のプログラムを操作するための情報であることを特徴とする請求項 1 乃至 5 のいずれか 1 項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 7】前記スキーマを構成するクラスに含まれるオブジェクトは、それぞれ外部から呼び出される場合のインタフェースとなる引数の個数が異なるメソッドを複数定義しており、

前記インタフェース定義は、前記オブジェクトのメソッドが有する引数の最大個数に対応する引数と、前記オブジェクトのメソッドが有する引数の最小個数に対応する引数の省略指定とによって定義されることを特徴とする請求項 1 乃至 6 のいずれか 1 項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項 8】オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得ステップと、

予め用意された第 1 のテンプレートを読み出し、該読み出した第 1 のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するためのプログラムを生成するプログラム生成ステップと、予め用意された第 2 のテンプレートを読み出し、該読み出した第 2 のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成ステップと、

10

20

30

40

50

3

インタフェース定義生成ステップと、を含むことを特徴とするオブジェクト指向データベース操作プログラム生成方法。

【請求項 9】 予め用意された第 3 のテンプレートを読み出し、該読み出した第 3 のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成ステップをさらに含むことを特徴とする請求項 8 に記載のオブジェクト指向データベース操作プログラム生成方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、オブジェクト指向データベースを操作するためのプログラム、特にオートメーションの機能を利用してオブジェクト指向データベースを操作するためのプログラムを生成するオブジェクト指向データベース操作プログラム生成システム及び方法に関する。

【0002】

【従来の技術】 インターフェース定義や分散ミドルウェアのクラスライブラリを用いたオブジェクトの開発は、分散システムを構築するための特有のプログラミング作業となる。この部分のプログラムを自動生成するアプローチ、或いはこの部分のプログラムの生成を支援しようとするアプローチは、従来よりかなりの数が提案されている。

【0003】 このようなアプローチとして、まず、特開平 4 - 1 1 1 0 2 2 号公報に記載のインタフェースプログラム自動生成装置（以下、従来例 1 という）が挙げられる。従来例 1 は、インタフェース定義、開発言語仕様、計算機種別データ表現の一般化／特殊化プログラムルーチン、通信プロトコル別プログラムルーチンの 4 つの情報源を基に、ネットワーク通信のための交換データ型定義とインタフェースプログラムとを自動生成する。

【0004】 また、上記のようなアプローチとして、特開平 8 - 3 6 4 8 6 号公報に記載の分散プログラム記述支援装置（以下、従来例 2 という）がある。従来例 2 は、特に CORBA に関して、開発言語によるオブジェクト記述から、構文解析、内部表現変換を経て、分散オブジェクトのインタフェース記述とインプリメンテーションとを記述する。

【0005】 さらに、上記のようなアプローチとして、特開平 8 - 2 6 3 2 7 7 号公報に記載のデータ操作プログラムの自動生成装置（以下、従来例 3）がある。従来例 3 は、データベース定義からデータの属性を解析する属性解析部の解析結果、及び型変換規則、データベースアクセステンプレート、補助関数生成規則に基づいて、既存の関係データベースのデータ操作を行うためのオブジェクト指向プログラムを生成する。

4

【0006】 上記のようなアプローチに従った実際の製品として、Visual C++（登録商標）にツールとして含まれる Developer Studio の機能である AppWizard 及び Class Wizard（いずれも商標）（以下、従来例 4 という）、並びにオブジェクト指向データベース管理システムである ObjectStore（登録商標）の拡張機能である PSE for ActiveX（以下、従来例 5 という）がある。

【0007】 従来例 4 では、AppWizard が開発の開始時点での初期設定入力からオブジェクト記述言語（以下、ODL という）のインタフェース定義とオートメーションオブジェクト（オートメーションによって操作可能なオブジェクトをいう。以下、同じ）の初版プログラムを自動生成する。ClassWizard がその後の開発言語によるプログラムの開発と並行して、ODL の記述とオートメーションオブジェクトのプログラムとを自動的に追加、削除、修正していく。

【0008】 従来例 5 では、ObjectStore が最初から持っているクラスライブラリの ODL 記述と、開発者が定義する ObjectStore 型記述ファイル（拡張子を ost として保存したファイル：以下、ost ファイルという）とを合わせて、ObjectStore の専用コンパイラでアプリケーションの ODL 記述と対応するオートメーションオブジェクトのプログラムとを生成する。

【0009】

【発明が解決しようとする課題】 しかしながら、上記の従来例には、次のような問題点があった。上記従来例 1 では、ネットワーク通信のための交換データ型定義とインタフェースプログラムとを自動生成するものの、インタフェース定義自体は、ユーザが与えなければならなかった。また、従来例 1 の装置は、既存のシステムを利用して構成されるものでないため、実際の製品を開発するためには、かなりの開発期間を要するという問題があった。しかも、インタフェース仕様に変更された場合には、プログラムをコンパイラの構文解析から作り直さなければならない、保守作業にも多くの時間を費やすこととなる問題があった。

【0010】 上記従来例 2 は、単一のオブジェクト指向プログラミング言語によるオブジェクト記述から、CORBA 仕様の分散オブジェクトのインタフェース記述とインプリメンテーション記述（ユーザ仕様言語）とを自動生成する機能を備える前提にしている。すなわち、オブジェクト変換プリプロセッサの構成に関するものであり、プログラムそのものを自動的に生成するものではない。しかも、インタフェース仕様に変更された場合には、プログラムをコンパイラの構文解析から作り直さなければならない、保守作業にも多くの時間を費やすこととなる問題があった。

【0011】 上記従来例 3 では、データベースアクセステンプレートに従って生成されるプログラムは、C++ などの特定の言語に依存するものとなる。このため、例

10

20

30

40

50

5

例えば、市販のアプリケーションプログラムが有するオートメーション通信機能を利用したツールを生かし切ることができないという問題点があった。また、インタフェース定義を自動的に生成するものではなく、インタフェース仕様が変更されて場合には、一からプログラムを作り直さなければならず、保守作業にも多くの時間を費やすこととなる問題点があった。

【0012】上記従来例4では、ClassWizardの機能は、AppWizardによって初版プログラムが生成されたプログラムに対してのみ、有効に働くようになっている。このため、AppWizardを使って生成していないプログラムに対して、ClassWizardの機能を十分に使うことができないという問題点があった。

【0013】上記従来例5では、ostファイルを作成するという別のインタフェース定義を行わなければならず、インタフェース定義そのものを自動生成することができない。このため、従来例3と同様に、市販のアプリケーションプログラムが有するオートメーション通信機能を利用したツールを生かし切ることができないという問題点や、インタフェース仕様が変更されて場合には、一からプログラムを作り直さなければならないという問題点があった。

【0014】さらに、データベース管理システムやAPI (Application Programing Interface) や、オブジェクト指向言語で書かれたサーバプログラムがオブジェクトの多相性をサポートしていても、クライアントからオートメーションを利用してアクセスした場合に、その多相性の機能を利用できる技術は、従来存在しなかった。また、既存のオートメーションの機能を利用してデータベースを操作できるようにするプログラムを生成する技術は、従来存在しなかった。

【0015】本発明は、また、開発・保守が容易なオブジェクト指向データベースを操作するためのプログラムを生成することができるオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【0016】本発明は、上記従来例の問題点を解消するためになされたものであり、既存のオブジェクト指向データベースを既存のオートメーション通信の機能を利用して操作するプログラムを、自動的に生成するオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【0017】本発明は、また、オブジェクトの多相性の機能を十分に利用することができるオブジェクト指向データベースを操作するためのプログラムを生成するオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【0018】

【課題を解決するための手段】上記目的を達成するため、本発明の第1の観点にかかるオブジェクト指向デー

6

タベース操作プログラム生成システムは、オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得手段と、前記スキーマを構成する各クラスに含まれるオブジェクトの操作に用いられるプログラムを生成するための第1のテンプレートを記憶する第1のテンプレート記憶手段と、前記第1のテンプレート記憶手段から前記第1のテンプレートを読み出し、該読み出した第1のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するための第1のプログラムを生成するプログラム生成手段と、前記プログラム生成手段が生成する前記スキーマを構成するクラス毎のプログラムが外部に公開すべきインタフェースを定義するためのインタフェース定義の生成に用いられる第2のテンプレートを記憶する第2のテンプレート記憶手段と、前記第2のテンプレート記憶手段から第2のテンプレートを読み出し、該読み出した第2のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成手段と、を備えることを特徴とする。

【0019】上記のオブジェクト指向データベース操作プログラム生成システムは、前記スキーマをクラスに含まれるオブジェクトを操作するための第1のプログラムと、インタフェース定義とのいずれをも自動的に生成することができる。このため、開発者がプログラムとインタフェース定義とのいずれか一方を与えたり、プログラムとインタフェース定義とを別々のシステムを用いて生成したりする必要がないので、オブジェクト指向データベースを操作するためのプログラムの開発が容易になる。また、インタフェース仕様の変更の場合にも、第1のプログラムを一から作り直さなければならないということがないので、プログラムの保守も容易である。

【0020】上記のオブジェクト指向データベース操作プログラム生成システムは、前記プログラム生成手段が生成した前記第1のプログラムと前記インタフェース定義生成手段が生成した前記インタフェース定義とをリンクすることによって、実行可能なモジュールを作り上げるリンク手段と、をさらに備えてもよい。

【0021】上記のオブジェクト指向データベース操作プログラム生成システムは、前記スキーマを構成する各クラスに含まれるオブジェクトを外部から操作するための外部操作情報を生成するための第3のテンプレートを記憶する第3のテンプレート記憶手段と、前記第3のテンプレート記憶手段から第3のテンプレートを読み出し、該読み出した第3のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの

10

20

30

40

50

少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成手段と、をさらに備えるものとしてもよい。

【0022】この場合、上記のオブジェクト指向データベース操作システムは、設定された情報によってインタフェースを介して外部から他のプログラムを操作するための第2のプログラムを実行させるプログラム実行手段と、前記外部操作情報生成手段が生成した外部操作情報を前記第2のプログラムに設定することによって、前記第2のプログラムが前記インタフェース定義手段によって定義されたインタフェース定義に対応するインタフェースを介して前記プログラム生成手段によって生成された前記第1のプログラムを操作することが可能となるように設定する設定手段と、を備えるものとすることができる。

【0023】これにより、上記のオブジェクト指向データベース操作プログラム生成システムは、例えば、インターネットなどを介して遠隔地から操作指令を送るようにしたクライアントサーバ型のシステム構成とする場合に、クライアント側で使用される第2のプログラムも、既存のプログラムを用いて容易に生成することができる。

【0024】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記プログラム生成手段は、例えば、前記スキーマを構成する各クラス毎に、プロパティの取得関数を生成する手段と、プロパティの設定関数を生成する手段と、メソッドを生成する手段とを備えるものとすることができる。

【0025】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記インタフェース定義によって定義されるインタフェースは、例えば、オートメーション通信を行うためのインタフェースとすることができる。この場合、前記外部操作情報は、オートメーション通信の機能を利用して前記プログラム生成手段が生成した前記第1のプログラムを操作するための情報とすることができる。

【0026】このような構成としたことにより、上記オブジェクト指向データベース操作プログラム生成システムは、既存のオートメーション通信のシステムを利用することができる。これにより、プログラムの開発、運用及び保守が容易になる。なお、オートメーション通信は、インターネットなどの通信回線を介して実現する場合に限らず、プロセス間通信の手法によって実現してもよい。

【0027】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記スキーマを構成するクラスに含まれるオブジェクトは、それぞれ外部から呼び出される場合のインタフェースとなる引数の個数が異なるメソッドを複数定義しているものであってもよ

い。この場合、前記インタフェース定義は、前記オブジェクトのメソッドが有する引数の最大個数に対応する引数と、前記オブジェクトのメソッドが有する引数の最小個数に対応する引数の省略指定とによって定義されるものとするを好適とする。

【0028】これにより、前記スキーマを構成するクラスに含まれるオブジェクトが多相性をサポートしている場合に、前記オブジェクトのメソッドに引数を渡すためのインタフェース仕様を統一することができるので、例えば、オートメーション通信を利用したような場合にも、オブジェクトの多相性の機能を利用することができる。

【0029】上記目的を達成するため、本発明の第2の観点にかかるオブジェクト指向データベース操作プログラム生成方法は、オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得ステップと、予め用意された第1のテンプレートを読み出し、該読み出した第1のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するためのプログラムを生成するプログラム生成ステップと、予め用意された第2のテンプレートを読み出し、該読み出した第2のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成ステップと、インタフェース定義生成ステップと、を含むことを特徴とする。

【0030】上記のオブジェクト指向データベース操作プログラム生成方法は、予め用意された第3のテンプレートを読み出し、該読み出した第3のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成ステップをさらに含むものとしてもよい。

【0031】

【発明の実施の形態】以下、添付図面を参照して、本発明の実施の形態について説明する。

【0032】図1は、この実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

【0033】図示するように、このシステムは、主としてサーバ1と、クライアント2と、プログラム自動生成装置3とを備える。サーバ1とクライアント2とは、インターネット5を介してオートメーション通信を行う。サーバ1とプログラム自動生成装置3とは、LAN回線などの何らかの通信回線を介してデータ通信を行う。プ

プログラム自動生成装置 3 は、プログラムテンプレートファイル 4 を参照する機能を有する。

【0034】サーバ 1 は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成され、データベース 1 1 と、API 1 2 と、サーバプログラム 1 3 と、オートメーションサーバ 1 4 と、リモートオートメーションマネージャ 1 5 との機能を含む。

【0035】データベース 1 1 は、オブジェクト指向データベースであり、クラス、クラスの属性、クラスのメソッドのそれぞれの名前と型との情報でスキーマ情報を定義する。

【0036】API 1 2 は、サーバプログラム 1 3、オートメーションサーバ 1 4、或いはプログラム自動生成装置 3 からデータベース 1 1 にアクセスするためのインタフェースである。

【0037】サーバプログラム 1 3 は、データベース 1 1 に定義されているスキーマクラスに含まれるオブジェクトのメソッドにメッセージを送るためのメソッドの実装などの処理を行うためのプログラムである。

【0038】オートメーションサーバ 1 4 は、後述するようにプログラム自動生成装置 3 によって生成されるオブジェクト指向型のプログラムである（以下、オートメーションサーバ 1 4 を構成するオブジェクトをオートメーションオブジェクトという）。オートメーションサーバ 1 1 は、データベース 1 1 のスキーマ情報を構成するクラスとその属性、メソッドを、オートメーションの機能によって外部のアプリケーションに公開する。リモートオートメーションマネージャ 1 5 は、既存のプログラムによって実現され、インターネット 5 を介してオートメーションサーバ 1 4 が公開するデータベース 1 1 のスキーマを構成するクラス（以下、スキーマクラスという）とその属性、メソッドを、クライアント 2 に提供する。

【0039】クライアント 2 は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成され、クライアントプログラム 2 1 とリモートオートメーションプロキシ 2 2 の機能を含む。

【0040】クライアントプログラム 2 1 は、オートメーションサーバ 1 4 が公開するデータベース 1 1 のスキーマクラスとその属性、メソッドを利用する外部アプリケーションとなり、サーバ 1 のデータベース 1 1 にアクセスする。リモートオートメーションプロキシ 2 2 は、既存のプログラムによって実現され、クライアントプログラム 2 1 がオートメーション通信でサーバ 1 が提供する機能を利用するために用いられる。

【0041】プログラム自動生成装置 3 は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成される。プログラム自動生成装置 3 は、データベース 1 1 から API 1 2 を介してそのデータベース 1 1 に定義されているスキーマ情報を取得し、取得したス

キーマ情報とプログラムテンプレートファイル 4 を基にして、オートメーション通信のためのオートメーションサーバ 1 4 のためのプログラムを自動生成する。生成されたプログラムは、さらにコンパイルされて、オートメーションサーバ 1 4 となる。

【0042】プログラムテンプレートファイル 4 は、後述するオートメーションオブジェクトのプログラムを生成するためや、後述する ODL 記述ファイルを生成するために用いられるテンプレートからなるファイルである。なお、プログラムテンプレートファイル 4 は、プログラム自動生成装置 3 が有する記憶装置内に記憶されていても、プログラム自動生成装置 3 の外部に設けられた記憶装置内に記憶されていてもよい。

【0043】以下、図 1 に示すクライアント 2 とサーバ 1 とに別れたデータベース操作システムを構築する手順について、図 2 のフローチャートを参照して説明する。

【0044】開発者は、まず、データベースマネジメントシステム（DBMS）を用いてデータベースのスキーマを定義し、データを登録することによってデータベース 1 1 を構築する（ステップ S 1 0 1）。

【0045】開発者は、次に、DBMS が用意する API 1 2 を用いてサーバプログラム 1 3 を作成し、データベース 1 1 のスキーマクラスを構成するオブジェクトにメッセージを送るためのメソッドを実装する（ステップ S 1 0 2）。

【0046】以上のステップ S 1 0 1 で構築されるデータベース 1 1、ステップ S 1 0 2 で作成されるサーバプログラム 1 3 は、既存のデータベース及びデータベースアプリケーションを利用するものとして行うことができる。

【0047】次に、プログラム自動生成装置 3 は、データベースのスキーマクラスのオートメーションサーバ 1 4 と、後述するクライアント 2 のセットアップ用に用いられるレジストリファイルとを自動生成する（ステップ S 1 0 3）。

【0048】以後は、サーバ 1 の側とクライアント 2 の側とに、処理が分かれる。サーバ 1 の側では、最初に、オートメーションサーバ 1 4 のレジストリ登録が行われる（ステップ S 1 0 4）。このレジストリ登録の詳細については、後述する。

【0049】サーバ 1 の側では、次に、リモートオートメーションマネージャ 1 5 を実装する処理が行われる（ステップ S 1 0 5）。このリモートオートメーションマネージャ 1 5 には、既存のオートメーション通信用のプログラムを利用することができる。また、既にサーバ 1 に実装されているものを利用することもできる。リモートオートメーションマネージャの実装が終了すると、サーバ 1 の側におけるシステムの構築が終了する。

【0050】クライアント 2 の側では、最初に、オートメーションサーバ 1 4 のレジストリ登録が行われる（ステップ S 1 0 6）。このレジストリ登録の詳細について

は、後述する。

【0051】クライアント2の側では、次に、リモートオートメーションプロキシ22を実装する処理が行われる(ステップS107)。このリモートオートメーションプロキシ22には、既存のオートメーション通信用のプログラムを利用することができる。また、既にクライアント2に実装されているものを利用することもできる。

【0052】クライアント2の側では、次に、オートメーションの機能をサポートする開発言語を用いて、インターネット5を介してオートメーションサーバ14を操作するための、クライアントプログラム21が作成される(ステップS108)。クライアントプログラム21が作成されると、クライアント2の側におけるシステムの構築の処理が終了する。

【0053】以下、図1のプログラム自動生成装置3によるオートメーションサーバ14及びレジストリファイルの生成の処理(ステップS103)の概要について、図3のフローチャートを参照して説明する。

【0054】プログラム自動生成装置3は、最初に、開発者が入力装置を操作することによって、初期設定値を入力する(ステップS201)。

【0055】ここで、入力する初期設定項目としては、例えば、①生成すべきオートメーションサーバ13の名前、②スキーマ情報を読み込むデータベース11の名前、③生成されたオートメーションサーバ13で公開されるクラスとなるオートメーションオブジェクトの名前、④サーバ1との間の通信プロトコル種別、⑤サーバ1のネットワークアドレス、などがある。

【0056】プログラム自動生成装置3は、次に、API12を介して登録されているクラス、クラスの属性、クラスのメソッドのそれぞれの名前と型の情報からなるデータベース11のスキーマ情報を読み込む(ステップS202)。ステップS202の処理を終了すると、プログラム自動生成装置3は、次に、ステップS203、S204、S206の処理を並行して行う。

【0057】ステップS203では、プログラム自動生成装置3は、プログラムテンプレートファイル4中のオートメーションオブジェクトに対応するテンプレートとなるファイルに、ステップS202で取得したスキーマ情報のコードを挿入することによって、オートメーションオブジェクトプログラムD201を生成する。

【0058】ステップS204では、プログラム自動生成装置3は、プログラムテンプレートファイル4中のODL記述ファイルのテンプレートとなるファイルに、ステップS202で取得したスキーマ情報のコードを挿入することによって、ODL記述ファイルD202を生成する。

【0059】ステップS203とS204の両方の処理が終了すると、プログラム自動生成装置3は、オートメ

ーションオブジェクトプログラムD201と、ODL記述ファイルD202と、サーバプログラムD203と、その他のオートメーションやサーバに必要なプログラムD204とを、それぞれプログラム開発言語環境に応じたコンパイラを用いてコンパイルし、さらにリンケージエディタを用いてリンクする(ステップS205)。これにより、オートメーションサーバ14のプログラムが生成される。

【0060】ここで、サーバプログラムD203は、図1のサーバプログラム13に相当するものである。その他のオートメーションサーバに必要なプログラムD204は、開発者が用いるサーバ1やプログラム開発言語環境に特有のプログラムである。

【0061】一方、ステップS206では、プログラム自動生成装置3は、クライアント2からサーバ2に接続するためのレジストリファイルD205を生成する。

【0062】以上の処理によって、オートメーションサーバ14のプログラムと、レジストリファイルD205とが生成されると、このフローチャートの処理を終了する。

【0063】以下、図3のフローチャートにおける各ステップの処理について、詳細に説明する。

【0064】まず、ステップS202のAPI12からスキーマ情報を読み込む処理について説明する。ここでは、プログラム自動生成装置3は、データベース11が稼働していることを条件として、データベース11に登録されているスキーマクラスをすべて取得することができるAPIやAPI群、或いはクラス毎にそのすべての属性の名前と型とアクセス識別子、さらにすべてのメソッドの名前と返却値の型と、すべての引数の型と、アクセス識別子とを取得することができるAPIやAPI群をAPI12として用いて、データベース11からスキーマ情報を読み込む。

【0065】次に、ステップS203のオートメーションオブジェクトプログラムのプログラムD201を生成する処理について説明する。ステップS203では、①プロパティ、②メソッド、③コンストラクタに分けて、オートメーションオブジェクトD201が生成される。

【0066】①のプロパティを生成するとき、プロパティの値の取得と設定を実現しなければならないので、取得関数と設定関数とが用意される。以下、それぞれの関数の生成手順について、図4と図5のフローチャートを参照して説明する。

【0067】なお、以下の説明において、Variant型とは、数値型、バイト型、文字列型、ブール型、日付型、通貨型、オブジェクト型などの数種類の型が定義されている共用体を持ち、その共用体で変数値の型を設定できる型のことを言う。

【0068】プロパティの取得関数を生成する場合は、図4のフローチャートに示すように、プログラム自動生

10

20

30

40

50

成装置 3 は、まず、戻り値を `Variant` 型で変数宣言し、その変数の型を上記のいずれかのうちの 1 つに設定する (ステップ S 3 0 1)。ここで、設定する変数の型は、ステップ S 2 0 2 で `API 1 2` を介してデータベース 1 1 から取得したスキーマ情報中のクラスの属性の型によって判断される。

【0069】例えば、クラスの属性の型が `char *` 型であったならば、変数の型は文字列型を表す `VT__BSTR` に設定される。或いは、クラスの属性の型が `int` 型であったならば、変数の型は、数値型のうちの整数型を表す `VT__I 2` (或いは長整数型 `VT__I 4`) に設定される。クラスの属性の型が他のクラスのオブジェクトを指すポインタ型であったならば、変数の型は、オブジェクト型を表す `VT__DISPATCH` に設定される。

【0070】プログラム自動生成装置 3 は、次に、DBMS の `API 1 2` を介してデータベース 1 1 にアクセスし、データベース 1 1 に格納されている目的とするオブジェクトの属性値を取得する (ステップ S 3 0 2)。

【0071】プログラム自動生成装置 3 は、次に、ステップ S 3 0 2 で取得した属性値を、ステップ S 3 0 1 で変数宣言した変数に設定した型で代入する。この代入の結果、返却された値がプロパティの取得関数のプロパティ値となる (ステップ S 3 0 3)。そして、プロパティの取得関数の生成の処理を終了する。

【0072】プロパティの設定関数を生成する場合は、図 5 のフローチャートに示すように、プログラム自動生成装置 3 は、最初に、関数内で有効なローカル変数を、関数言語で指定される型、すなわち対象とする属性の型で宣言する (ステップ S 4 0 1)。

【0073】プログラム自動生成装置 3 は、次に、ステップ S 4 0 1 で指定された `Variant` 型の引数を取り出し、それをステップ S 4 0 1 で宣言したローカル変数に代入する (ステップ S 4 0 2)。

【0074】プログラム自動生成装置 3 は、次に、`API 1 2` を介してデータベース 1 1 にアクセスし、データベース 1 1 に格納されている目的とするオブジェクトの属性値に、ローカル変数の値を代入 (またはコピー) する (ステップ S 4 0 3)。

【0075】プログラム自動生成装置 3 は、さらに、`API 1 2` を介してデータベース 1 1 にオブジェクトの属性を更新したことを通知し、これによってプロパティ値をデータベース 1 1 に格納されている目的とするオブジェクトの属性に反映させて、更新させるためのプログラムとして必要となるコードを生成する (ステップ S 4 0 4)。そして、プロパティの設定関数の生成の処理を終了する。

【0076】②のオートメーションオブジェクトのメソッドの生成手順について、図 6 のフローチャートを参照して、詳しく説明する。

【0077】プログラム自動生成装置 3 は、まず、関数

内で有効となる `Variant` 型のローカル変数を宣言し、メソッドからの戻り値の型に対応する `Variant` 型の型を設定する (ステップ S 5 0 1)。

【0078】プログラム自動生成装置 3 は、次に、各引数毎に対応する開発言語仕様の型のローカル変数を用意し、その変数にそれぞれの `Variant` 型の引数の値を代入する (ステップ S 5 0 2)。

【0079】プログラム自動生成装置 3 は、次に、各引数が代入されたローカル変数の値をパラメータ値として、`API 1 2` を介してデータベース 1 1 に定義されているクラスのメソッドを実行させる (ステップ S 5 0 3)。

【0080】プログラム自動生成装置 3 は、次に、ステップ S 5 0 3 でのメソッドの実行の結果、値が変わったローカル変数の値を、対応する型で `Variant` 型の引数の値に代入し、メソッドに実行による値の変化を反映させる (ステップ S 5 0 4)。

【0081】プログラム自動生成装置 3 は、さらに、ステップ S 5 0 3 でのメソッドの実行後に返却された値を、ステップ S 5 0 1 で用意した `Variant` 型のローカル変数に、対応する型で代入し、それをオートメーションメソッドの返却値とする (ステップ S 5 0 5)。そして、このフローチャートの処理を終了する。

【0082】③のオートメーションオブジェクトのコンストラクタの生成手順について、詳しく説明する。

【0083】オートメーションオブジェクトは、データベース 1 1 のスキーマクラスを代理するものなので、プログラム自動生成装置 3 は、オートメーションオブジェクトのクラス定義に、そのスキーマクラスの参照型 (ポインタ型) のオブジェクトを保持するメンバ変数の宣言を自動生成する。

【0084】そして、プログラム自動生成装置 3 は、スキーマクラスの参照型のオブジェクトを生成し、生成したオブジェクトにメンバ変数を代入する文を挿入することによって、オートメーションオブジェクトのコンストラクタを生成する。

【0085】オートメーションオブジェクトのコンストラクタは、クライアントプログラム 2 1 からオートメーションオブジェクトのオブジェクト型変数の宣言があったときに呼び出される。従って、クライアント 2 の側で扱うオブジェクト型の変数は、スキーマクラスの参照型のオブジェクトということになる。

【0086】もっとも、スキーマクラスの参照型は、データベース 1 1 のオブジェクトを指すポインタであるので、オートメーションオブジェクトのコンストラクタが呼ばれるときに、スキーマクラスのオブジェクトが生成される訳ではない。つまり、オートメーションオブジェクトのコンストラクタとスキーマクラスのコンストラクタとは異なるものであり、スキーマクラスのオブジェクトの生成とそれに伴うコンストラクタの呼び出しは、別

にメンバ関数を用意することによって対応が図られている。

【0087】次に、ステップS204のオートメーションサーバのODL記述ファイルD202を生成する処理について、図7のフローチャートを参照して説明する。この処理は、図3のステップS202で、API12を介してデータベース11から読み込んだスキーマクラスのすべての情報に対して適用される。

【0088】プログラム自動生成装置3は、まず、COMのAPIである“CoCreateGuidを用いて、各クラス毎にディスパッチインタフェース用のIDを割り当て、ODL記述ファイルに記述する（ステップS601）。

【0089】プログラム自動生成装置3は、次に、ディスパッチインタフェースの名前としてスキーマクラスの名前に接頭語“I”を付け足した名前を用いて、ODL記述ファイルの“dispinterface”文を記述する。“dispinterface”文では、“properties”と“methods”とを記述するが、ここでは、スキーマクラスの属性名とメソッドとがそのまま用いられる。また、返却値の型は、すべてVariant型で、引数の型はすべてVariantのアドレスの型で宣言される。また、“properties”と“methods”とのDISPIDには、1から通し番号が付けられる（ステップS602）。

【0090】プログラム自動生成装置3は、次に、“CoCreateGuid”を用いて各クラス毎にオートメーションオブジェクト用のIDを割り当て、ODL記述ファイルに記述する（ステップS603）。

【0091】プログラム自動生成装置3は、さらに、“dispinterface”文と1対1で対応するように、同じスキーマの名前で“coclass”文を記述する。“coclass”文の中には、その“coclass”の名前に接頭語“I”を付けた“dispinterface”文をサポートする宣言を記述する（ステップS604）。そして、このフローチャートの処理を終了する。

【0092】次に、ステップS206のクライアントセットアップ用のレジストリファイルD205を生成する処理について、説明する。

【0093】ステップS206で生成されるレジストリファイルD205は、例えば、Windows95やWindowsNT（いずれも登録商標）のレジストリエディタで、クライアント2のレジストリなどにオートメーションサーバ14の情報を登録するために用いられる。

【0094】レジストリに登録すべきオートメーションサーバ14の情報には、オートメーションオブジェクト毎に、①オートメーションサーバの名前、②オートメーションオブジェクトの名前、③クラスID、④通信プロトコル種別、⑤ネットワークアドレス（IPアドレス）、⑥クライアントプログラム21の実行時に必要なダイナミックリンクライブラリなどが含まれる。

【0095】このうち、①～⑤は、ステップS201で

初期設定入力された情報と、ステップS202で読み出されたスキーマ情報とから取得される。⑥は、固定的なものであり、プログラムテンプレートファイル4中のレジストリファイルのテンプレートファイルに記述することによって用意される。

【0096】プログラム自動生成装置3は、プログラムテンプレートファイル4中のレジストリファイルのテンプレートとなるファイルに、これらの情報のコードを挿入することによってレジストリファイルD205を生成する。

【0097】以下、図3のフローチャートにおけるサーバ1の側でのオートメーションサーバ14のレジストリ登録（ステップS105）、及びクライアント2の側でのオートメーションサーバ14のレジストリ登録と接続設定（ステップS107）の手順について、詳しく説明する。

【0098】ステップS105において、レジストリ登録される情報は、図2のステップS206で生成されたレジストリファイルD205に含まれる情報のうちの①オートメーションサーバの名前、②オートメーションオブジェクトの名前、③クラスIDである。これらの情報は、通常、オートメーションサーバ14の起動時に実行されるレジストリ登録用のAPIによってオートメーションサーバ14のレジストリに登録することができる。

【0099】ステップS107において、図2のステップS206で生成されたレジストリファイルD205に含まれる情報が、例えば、Windows95やWindowsNTのレジストリエディタに読み込ませて、クライアントプログラム21のレジストリに登録される。なお、通信プロトコル種別やネットワークアドレスなどのサーバ1への接続に関する設定は、レジストリファイルD205をクライアント21のレジストリに登録した後でも、例えば、リモートオートメーション接続マネージャなどを用いて変更することが可能である。

【0100】以下、上記のようにして構築したシステムで実行されるデータベースアプリケーションプログラムで、オブジェクトの多相性が如何にして実現されているかについて、詳しく説明する。

【0101】この実施の形態のシステムにおいて、多相性プログラムの実現方法は、オートメーションオブジェクトに多相性を実現すべきオブジェクトのクラス名を記録させる属性を追加し、プログラムの実行時には、オートメーションオブジェクトは、その属性から得たクラス名によって当該オブジェクトのメソッドを呼び出すという考え方に基づいている。

【0102】また、オートメーションサーバ14が公開するオートメーションオブジェクトの引数の型をすべてVariant型とし、そのVariant型の変数値の型をオートメーションサーバ14が判別することによって、適切なメソッドを呼び出している。以下、メソッ

10

20

30

40

50

ドの多重定義の場合と、仮想関数の場合に分けて、多相性の実現方法を詳しく説明する。

【0103】オブジェクトの多相性を実現するために、オートメーションオブジェクトは、引数の型及び／または個数が異なるメソッドを複数定義している。ここで、メソッドの引数の最小個数をm（自然数）、最大個数をM（自然数）とすると、ODL記述ファイルは、①Variant型の戻り値、②メソッド名、③M個のVariant型の引数、④M-m個のoptionalパラメータを指定したVariant型の引数によってメソッドを定義する。

【0104】④のoptionalパラメータは、オートメーションメソッドの呼び出し時に、パラメータによって対応する引数の値を省略すべきことを示すものである。但し、optionalパラメータによる指定は、すべてVariant型の引数でなければならないという制限がある。

【0105】次に、仮想関数の場合、仮想関数は、サーバ1のが側の開発言語や、データベース11にアクセスするためのAPI12がサポートしているので、仲介者となるオートメーションサーバ14は、クライアントプログラム21がどのオブジェクトのメソッドを呼び出したのかをサーバプログラム13に渡せばよいこととなる。

【0106】仮想関数の処理においては、“virtual”で定義しているメンバ変数をもつ親クラス（スーパークラス）のオブジェクトと、その親クラスから派生したクラス（派生クラス：サブクラス）のオブジェクト間の実行時の切換が重要となる。

【0107】仮想関数の運用において、まず、親クラスのオブジェクトへのポインタを宣言し、そのポインタに対してメンバ関数の呼び出しが記述される。そして、実行時に、そのポインタが指すオブジェクトの実体（インスタンス）が、親クラスのオブジェクトであるか派生クラスのオブジェクトかをシステムが判断し、そのどちらか一方の正しいオブジェクトのメソッドが呼び出される。

【0108】この実施の形態のシステムにおいては、クライアント2の側で用いられるオートメーションオブジェクトは、オートメーションサーバ14においてスキーマクラスのオブジェクトを指すポインタとして実装されている。従って、オートメーションサーバ14が仮想関数の処理を正しく中継するためには、クライアント2の側で親クラスのオブジェクト型変数の実体の実行時に派生クラスのオブジェクト型変数に置き換わったときに、オートメーションサーバ14も同様のポインタの置き換えを行えばよい。

【0109】この実施の形態のシステムでは、このポインタの置き換えについて、ポインタ型の代入演算子の代わりとなるメンバ関数を親クラスのオートメーションオ

ブジェクトに用意することによって、対応している。このメンバ関数は、派生クラスのオブジェクト変数を引数にとり、そのメンバ関数が呼ばれる親クラスのオブジェクトへのポインタを、引数である派生クラスのオブジェクトのポインタに置き換える処理を行うものである。

【0110】この処理によって、クライアントプログラム21で親クラスのオブジェクトからのメンバ変数の呼び出しが記述されていても、そのオブジェクトの実体を実行時に置き換えることが可能となり、オートメーションサーバ14において、仮想関数の振る舞いを実現することが可能となる。

【0111】以下、上記のようにして構築したシステムにおいて、クライアントプログラム21からデータベース11へアクセスするための処理について、図8のシーケンス図を参照して説明する。

【0112】なお、ここで、クライアントプログラム21とオートメーションサーバ14との間の通信は、リモートオートメーションプロキシ22とリモートオートメーションマネージャ15とを用い、インターネット5を介して行われる。また、オートメーションサーバ14或いはサーバプログラム13とデータベース11との間のデータのやり取りは、API12を介して行われる。

【0113】以下、図8のシーケンス図に示す処理において、オートメーションサーバ14が実行する処理の詳細について、図9のフローチャートを参照して説明する。

【0114】オートメーションサーバ14は、まず、クライアントプログラム21から呼び出されたメソッドの引数の個数をチェックし、引数が1つ以上あるかどうかを判定する（ステップS701）。

【0115】ステップS701で引数が1つ以上ない、すなわち引数がないと判定されたときは、オートメーションサーバ14は、引数なしのメソッドを呼び出す（ステップS702）。そして、このフローチャートの処理を終了する。

【0116】ステップS711で引数が1つ以上あると判定されたときは、オートメーションサーバは、メソッドの第1引数の値からすべての引数の値の処理が終了するまで、ステップS703-S703'のループの処理を繰り返す。もっとも、後述するように、すべての引数の値の処理を終了する前に、このループから抜け出す場合はある。

【0117】ステップS703-S703'のループの処理では、オートメーションサーバ14は、まず、メソッドの引数がoptional指定されているかどうかを判定する（ステップS704）。

【0118】ステップS704でメソッドの引数がoptional指定されていると判定されたときは、オートメーションサーバ14は、optional指定の値が設定されているかどうかを判定する（ステップS70

10

20

30

40

50

5)。

【0119】ステップS705でoptional指定の値が設定されていないと判定されたときは、このループを抜けて、後述するステップS711に進む。ステップS705でoptional指定の値が設定されていると判定されたときは、ステップS706に進む。また、ステップS704でメソッドの引数がoptional指定されていないと判定されたときも、後述するステップS706に進む。

【0120】ステップS706では、オートメーションサーバ14は、Variant型引数の値がどのような型であるかをチェックする。オートメーションサーバ14は、さらにステップS706でチェックした引数の型がオブジェクト型であるかどうかを判定する(ステップS707)。

【0121】ステップS707で引数の型がオブジェクト型であると判定されたときは、オートメーションサーバ14は、Variant型のオブジェクト型の変数のクラス名を、各クラスで実現されているオブジェクト自身にそのオブジェクトのクラス名を記録しておくリフレクションの機能を用いて、チェックする(ステップS708)。

【0122】オートメーションサーバ14は、さらに、クラス名に対応するスキーマクラスのオブジェクトを宣言し、宣言したオブジェクトに、ステップS707で判定されたオブジェクト型の引数を代入する(ステップS709)。ステップS709の処理が終了すると、メソッドに次の引数があればその引数についてのループ(S703-S703')の処理に、次の引数があれば、ループ(S703-S703')を抜けて、ステップS711の処理に進む。

【0123】ステップS707で引数の型がオブジェクト型でないと判定されたときは、オートメーションサーバ14は、ステップS706でチェックした型に対応する開発言語仕様の型で変数宣言をし、宣言した変数に引数の値を代入する(ステップS710)。ステップS710の処理が終了すると、メソッドに次の引数があればその引数についてのループ(S703-S703')の処理に、次の引数があれば、ループ(S703-S703')を抜けて、ステップS711の処理に進む。

【0124】ステップS711では、オートメーションサーバ14は、以上の処理で準備された変数やオブジェクトを引数として、多重定義されたメソッドを呼び出す。そして、このフローチャートの処理を終了する。

【0125】以上説明したように、プログラム自動生成装置3は、データベース11に定義されているスキーマクラスの情報を取得し、取得した情報のうちの適切なものをテンプレートファイル4のうちのオートメーションサーバ生成用のものに挿入することによって、オートメーションサーバ14を生成することができる。また、取

得した情報のうちの適切なものをテンプレートファイル4のうちのODL記述ファイル生成用のものに挿入することによって、ODL記述ファイルを生成し、これにより、インタフェース定義を定めることができる。さらに、取得した情報からレジストリ登録に使用する情報からなるレジストリファイルも生成することができる。

【0126】従って、この実施の形態のプログラム自動生成装置3によれば、オブジェクト指向のデータベース11を操作するためのプログラムの生成が容易になる。また、例えば、インタフェース仕様に変更された場合などにも、プログラムを一から作り直す必要がないので、プログラムの運用、保守も容易になる。

【0127】しかも、データベース11、クライアントプログラム21、或いはオートメーション通信を行うためのリモートオートメーションマネージャ15及びリモートオートメーションプロキシ22には、既存のシステムを利用することができるので、プログラムの開発、さらにはテストを容易に行うことができる。

【0128】さらに、optionalパラメータの導入によって、クライアント2側のクライアントプログラム21とサーバ1側のオートメーションサーバ14とのオートメーション通信におけるインタフェース仕様を統一することができ、なおかつデータベース11に登録されているオブジェクトの多相性の機能を十分に利用することができる。

【0129】上記の実施の形態では、プログラム自動生成装置3は、サーバ1とは別個の情報処理装置によって構成されていた。しかしながら、サーバ1がプログラム自動生成装置3の機能を含むものとして、サーバ1とプログラム自動生成装置3とを同一の情報処理装置によって構成することもできる。この場合において、プログラムテンプレートファイル4は、サーバ1内の記憶装置に記憶することも外部の記憶装置に記憶することもできる。

【0130】上記の実施の形態では、オートメーションオブジェクトプログラム、ODL記述及びレジストリファイルは、すべて同一のプログラム自動生成装置3によって生成されていた。しかしながら、これらは、それぞれ別々の情報処理装置によって生成し、コンパイルした後に、リンクしてもよい。

【0131】上記の実施の形態では、インターネット5を介して接続されたサーバ1とクライアント2とからなるクライアントサーバシステムで本発明のプログラムの開発・実行環境を実現した場合について説明した。しかしながら、本発明はこれに限られない。例えば、イントラネット環境でのクライアントサーバシステムを本発明のプログラムの開発・実行環境とすることもできる。

【0132】また、クライアント(マシン)2とサーバ(マシン)1とが別個に存在するクライアントサーバシステムではなく、図10に示すようなスタンドアローン

10

20

30

40

50

型のコンピュータシステム 6 も、本発明のプログラムの開発・実行環境とすることができる。この場合、クライアントプログラム 2 1 のプロセスとコンピュータシステム 6 上にオートメーションサーバ 1 4 のサーバプロセスとが存在することとなり、クライアントプログラム 2 1 とオートメーションサーバ 1 4 とがプロセス間通信の手法を用いてオートメーション通信を行う。

【0 1 3 3】上記の実施の形態では、COM/DCOM を基礎とするオートメーション通信を例として説明した。しかしながら、本発明におけるオートメーション通信は、COM/DCOM を基礎とするものに限るものではなく、本発明の趣旨を逸脱しない限りにおいて、他の仕様を基礎とするものにも適用することができる。

【0 1 3 4】

【実施例】以下、上記の実施の形態のデータベースアプリケーションプログラムの開発・実行環境を具体的なシステムで実現した場合の実施例について、プログラムリスト（ソースリスト）を参照しながら説明する。

【0 1 3 5】この実施例のシステムでは、データベース 1 1 としてオブジェクト指向データベース PERC IO（登録商標）、サーバプログラム開発言語として Visual C++（登録商標）、クライアントプログラム開発言語として Visual Basic（登録商標）を用いている。

【0 1 3 6】1) オートメーションオブジェクトのクラス
オートメーションオブジェクトのヘッダファイルの一部となるプログラムテンプレートファイル 4 のプログラムリストの一例を以下に示す。この例では、“@@”で囲まれる文字列が、オートメーションサーバ 1 4 が API を介してデータベース 1 1 から取得するスキーマクラスの情報が埋め込まれる個所である。従って、以下の“@@CLASS@@”は、クラス名に置き換えられる。

【0 1 3 7】また、“dRef”は、すべてのユーザ定義スキーマクラスの基本クラスとなっている。オートメーションオブジェクトのクラスは、“dRef”を継承して定義されている。“d_Extent”“d_Set”、“d_List”は、PERC IO がシステムとして予め用意しているテンプレート型のスキーマクラスである。“d_Ref_Any”は、PERC IO がシステムとして予め用意しているデータベース中の任意のオブジェクトへのポインタを保持する変数の型である。

【0 1 3 8】オートメーションオブジェクトのクラスは、これらのシステムが用意するスキーマクラスのオブジェクトを生成するメンバ関数“createextent()”、“createset()”、“createlist()”の関数宣言と、自分自身を指すポインタを生成するメンバ関数“create_newref()”の関数宣言とを定義する。

【0 1 3 9】なお、この実施例で示されている、接頭語“C”で始まるクラスは、オートメーションオブジェク

トのクラスを表し、接頭語“C”をとったクラス名のスキーマクラスと 1 対 1 で対応する。また、接頭語“d_”で始まるクラスは、PERC IO がシステムとして予め用意するスキーマクラスであり、アンダースコアのない接頭語“d”で始まるクラスは、PERC IO がシステムとして用意するクラスと 1 対 1 で対応するオートメーションクラスを表している。

【0 1 4 0】

```
class C@@CLASS@@ : public dRef
{
    DECLARE_DYNCREATE(C@@CLASS@@)
    C@@CLASS@@();
public:
    d_Extent<d_Ref_Any>* createextent();
    d_Set<d_Ref_Any>* createset();
    d_List<d_Ref_Any>* createlist();
    LPDISPATCH create_newref();
```

【0 1 4 1】2) オートメーションオブジェクトの基本クラスのプログラム

オートメーションオブジェクトのクラス“dRef”のヘッダファイルのプログラムリストの一部を以下に示す。

“dRef”は、すべてのユーザ定義スキーマクラスに共通の基本クラスであり、固定的なファイルとしてプログラムテンプレートファイル 4 に用意されている。

【0 1 4 2】

```
class dRef : public CCmdTarget
{
    DECLARE_DYNCREATE(dRef)
    dRef();
public:
    d_Ref_Any m_classref;
    CString m_classname;
public:
    virtual d_Extent<d_Ref_Any>* createextent();
    virtual d_Set<d_Ref_Any>* createset();
    virtual d_List<d_Ref_Any>* createlist();
    virtual LPDISPATCH create_newref();
protected:
    virtual dRef();
};
```

【0 1 4 3】オートメーションオブジェクトのテンプレートファイルで定義されているメンバ変数は、すべて“dRef”クラスで、仮想関数で定義されている。また、属性の定義としては、オートメーションオブジェクトが代理として振る舞っている PERC IO のオブジェクトへのポインタを保持する“m_classref”、オートメーションオブジェクト自身のクラス名を保持する“m_classname”がある。“m_classref”は、オートメーションオブジェクトのコンストラクタの説明の部分で述べたスキーマクラスの参照型のオブジェクトを保持するメンバ変

数である。また、“m_classname”は、多重定義の説明の部分で述べたオブジェクトのクラス名を記憶させるメンバ変数である。

【0144】3) オートメーションオブジェクトのプロパティのプログラム

図4のプロパティの取得関数の生成手順を実現するプログラムリストを以下に示す。このプログラムは、スキー

```
VARIANT C@@CLASS@::_Get@@VAR@@()
{
    VARIANT vaResult;
    VariantInit(&vaResult);
    V_VT(&vaResult) = VT_I4;
    if (!m_classref)
        return vaResult;
    V_I4(&vaResult) = GETOBJ(m_classref)->@@ORIGVAR@@;
    return vaResult;
}
```

【0146】図5のプロパティの設定関数の生成手順を実現するプログラムリストを以下に示す。このプログラムは、スキーマクラスのchar*型の属性値を取得するプロパティの取得関数を示すものである。ここで、最後の行にある“GETOBJ”はマクロであり、“m_classref”のオブジェクトの参照型に展開される。また、“mar

```
void C@@CLASS@::_Set@@VAR@@(const VARIANT FAR& newValue)
{
    CString c_@@VAR@@;
    if (!m_classref)
        return;
    c_@@VAR@@ = CString(V_BSTR(&newValue));
    strcpy(GETOBJ(m_classref)->@@ORIGVAR@@, c_@@VAR@@);
    GETOBJ(m_classref).mark_modified();
}
```

【0148】4) オートメーションオブジェクトのメソッドのプログラム

図6のオートメーションメソッドの生成手順を実現するプログラムリストを以下に示す。このプログラムは、ユーザが定義したスキーマクラス“Employee”を代理するオートメーションオブジェクトのクラス“CEmployee”のメソッド“getdata”を実装するものである。スキーマクラスのメソッド名は、“get_data”である。また、“CopyToVARIANT”は、グローバル関数であり、API呼び出しの結果置き換わったパラメータ“n_param0”の値を、オートメーションメソッドのVariant型の引数“param0”にコピーする。

【0149】

マクラスのint型の属性値を取得するプロパティの取得関数を示すものである。この例では、“@@ORIGVAR@@”がスキーマクラスの情報の属性名と置き換わる。また、“@@VAR@@”は、その属性の中でアンダースコアを取り除いた文字列と置き換わる。

【0145】

k_modified()”は、PERC IOが用意するシステムのメンバ関数であり、プロパティ設定においてオブジェクトの属性値が変更されたことをデータベース11に通知して、データベース11の内容を更新する。

【0147】

```
VARIANT CEmployee::getdata(VARIANT FAR* param0)
{
    VARIANT vaResult;
    VariantInit(&vaResult);
    char *n_param0 = NULL;
    if (AllocAndCopyCharParam(&n_param0, param0)) {
        return vaResult;
    }
    GETOBJ(m_classref)->get_data(n_param0);
    CopyToVARIANT(n_param0, param0);
    if (n_param0)
        free(n_param0);
    return vaResult;
}
```

【0150】5) サーバプログラム

スキーマクラス“Employee”のメンバ関数get_dataの実装例を以下に示す。このプログラムは、図1のサーバプログラム13に対応し、また、このプログラムの作成

は、図 2 のステップ S 1 0 2 の処理に相当する。

【0151】

```
void Employee::get_data(char* data)
{
    char tempstr[1024];
    sprintf(tempstr, "%s#t%s#t%d#t%s#t%d#r#n",
        (char*)name,
        (char*)birth_day,
        get_age(),
        (char*)section,
        salary);
    strcpy(data, tempstr);
}
```

【0152】 6) ODL 記述ファイル

ODL 記述ファイルを生成するためのプログラムテンプレートファイル 4 のうちのテンプレートの例を以下に示す。ここでは、“dispinterface” 文の “properties” までを記述するテンプレートファイルの例を示す。以下の例は、図 7 の ODL 記述の生成手順において用いられる。

【0153】

```
dispinterface I@@CLASS@@
{
    properties:
        //{AFX_ODL_PROP(C@@CLASS@@)
```

【0154】 この後に、スキーマクラスの属性が挿入される。“dispinterface” 文の “method” までを記述するテンプレートファイルを以下に示す。

【0155】

```
[ uuid(5794F9E6-3A54-11d1-BB17-00004C95A764) ]
dispinterface IEmployee
{
    properties:
        //{AFX_ODL_PROP(CEmployee)
        [id(1)] VARIANT name;
        [id(2)] VARIANT birthday;
        [id(3)] VARIANT section;
        [id(4)] VARIANT salary;
        //{AFX_ODL_PROP
    methods:
        //{AFX_ODL_METHOD(CEmployee)
        [id(5)] VARIANT create(VARIANT * db_obj, VARIANT* param0, VARIANT*
param1, VARIANT* param2, VARIANT*param3);
        [id(6)] VARIANT setsalary(VARIANT* param0);
        [id(7)] VARIANT getsalary();
        [id(8)] VARIANT getdata(VARIANT* param0);
        [id(9)] VARIANT getall(VARIANT* param0);
        [id(10)] VARIANT remove();
        //{AFX_ODL_METHOD
```

//}}AFX_ODL_PROP

methods:

://{AFX_ODL_METHOD(C@@CLASS@@)

【0156】 この後に、スキーマクラスのメソッドが挿入される。“dispinterface” 文の最後の部分の生成に用いられるテンプレートファイルを以下に示す。

【0157】

//}}AFX_ODL_METHOD

};

10

// Class information for C@@CLASS@@

【0158】 “coclass” 文を記述するためのテンプレートファイルを以下に示す。

【0159】

```
coclass @@CLASS@@
{
    [default] dispinterface I@@CLASS@@;
};
```

【0160】 上記の図 7 のフローチャートで示した手順に従ってこの実施例のプログラム生成装置 3 が生成し

20

た、オートメーションオブジェクトのクラス “CEmployee” の ODL 記述ファイルの例を以下に示す。この ODL 記述ファイルをコンパイルし、ライブラリを作成して公開すれば、オートメーションオブジェクトを外部と連携することが可能となる。

【0161】

```
};
// Class information for CEmployee
[ uuid(5794F9E7-3A54-11d1-BB17-00004C95A764) ]
coclass CEmployee
{
    [default] dispinterface IEmployee;
}
```

【0162】7) クライアントセットアップ用レジストリファイル

図3のステップS206で生成されるレジストリファイルD205の、この実施例における生成例を以下に示す。この例では、オートメーションサーバの名前が、“BookAX”、オートメーションオブジェクトの名前が“CEmployee”、オートメーションサーバ“BookAX”のIPアドレスが“133.207.62.173”と設定されている。また、オートメーションオブジェクト“CEmployee”のCLSIDは、“5794F9E7-3A54-11d1-BB17-00004C95A764”に設定されており、ODL記述ファイル中の“coclass”文の“CEmployee”の“uuid”と一致する。なお、このレジストリファイルの内容は、図2のステップS1

【0163】

```
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}]
@="BookAX. CEmployee"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥LocalServer32]
@="BookAX. EXE"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥_LocalServer32]
@="BookAX. EXE"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥ProgId]
@="BookAX. CEmployee"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥InprocHandler32]
@="ole32. dll"
[HKEY_CLASSES_ROOT¥BookAX. CEmployee]
@="BookAX. CEmployee"
[HKEY_CLASSES_ROOT¥BookAX. CEmployee¥CLSID]
@="{5794F9E7-3A54-11d1-BB17-00004C95A764}"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥InprocServer32]
@="autprx32. dll"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥NetworkAddress]
@="133. 207. 62. 173"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥ProtocolSequence]
@="ncacn_ip_tcp"
```

【0164】8) クライアントプログラム

Visual Basicによって生成した、この実施例のクライアントプログラム21の例を以下に示す。これは、図2のステップS108で生成されるものである。この例では、オートメーションサーバ14には、“BookAX”と“CPPAPI”という2つの名前が用いられており、ユーザが定義したスキーマクラスを代理するオートメーションサーバには“BookAX”が、PERC IOが用意するシステムのスキーマクラスを代理するオートメーションサーバには“CPPAPI”が用いられている。

【0165】“Dim”ステートメントでVariant型の変数を宣言し、それに続く“CreateObject”関数でオートメーションオブジェクトを参照し、“Set”ステートメントでVariant型の変数にオブジェクト型として保持させる。この2つのVariant型の変数によって、PERC IOのスキーマクラスのオブジェクトへの参照関係も解除される。

【0166】また、プログラムの後半で用いられている“Grid1”や“Text1”などで表されている“ActiveX”コントロールは、グリッドコントロールとテキストボックスコントロールとを用いて、データベース11の検索結果を表示するときに用いられている。

30 【0167】

```
Private Sub printemp()
    Dim emp As Variant
    Set emp = CreateObject("BookAX. CEmployee")
    Dim ext As Variant
    Set ext = CreateObject("CPPAPI. dExtent")
    ext.init emp
    Dim itr As Variant
    Set itr = ext.createiterator
    Dim str As Variant
    40 Dim sum As Variant
    Dim i As Integer
    Do While itr.Next(emp)
        str = String(64, " ")
        emp.GetData str
        sum = sum + str
        i = i + 1
        Set str = Nothing
    Loop
    Grid1.Clip = sum
    50 Text1.Text = i
```

```
Set emp = Nothing
Set ext = Nothing
Set itr = Nothing
End Sub
```

【0168】9) 多重定義

スキーマクラスのメソッド“get_data”は、引数が最小で1個、最大で3個の多重定義されているメソッドであるとする。この場合、ODL記述ファイルには、optional指定を用いてオートメーションメソッド“getdata”を記述することとなるが、この実施例において、プログラムは以下のように記述される。

【0169】以下のようなoptional指定をODL記述ファイルに記述することによって、オートメーションサーバ14は、クライアントプログラム21から渡された引数の型をVariant型からデータベース1

```
Variant CPerson::cast(VARIANT FAR* param0)
{
    dRef *src_dRef =
        dynamic_cast<dRef*>(CCmdTarget::FromIDispatch(V_DISPATCH(param0)));
    if(src_dRef != NULL) {
        d_Ref<Person> c_person;
        c_person = (d_Ref<Person>)src_dRef->m_classref;
        m_classref = (d_Ref_Any)c_person;
    }
}
```

【0173】以下、この実施例のシステムにおけるデータベースアプリケーションプログラムの開発プロセスについて、図11のフローチャートを参照して説明する。

【0174】まず、開発者は、PERCIO（データベース11）のスキーマを設計し、データベースの構築を行う（ステップS801）。設計したデータベースのスキーマは、拡張及び／または変更をすることが可能である（ステップS802）。さらに、開発者は、各スキーマクラス毎にメソッドを実装する（ステップS803）。ここで実装したメソッドが、上記のサーバプログラム13の一部となる。なお、以上のステップS803までで示したシステムの構築を行う代わりに、既存のシステムを用いることも可能である。

【0175】以後、開発プロセスは、システムの開発がサーバ1側とクライアント2側とで並行して進めることが可能となる。サーバ1の側では、後述するステップS804～S810のすべての開発プロセスを実行することが可能となる。一方、クライアント2の側では、後述するステップS805とS806、及びS808～S810までの開発プロセスを実行することが可能となる。

【0176】次に、開発者は、上記の実施の形態で示したプログラム自動生成装置3を用いて、データベース11に登録されているスキーマクラスに対応したオートメーションサーバ14を生成する（ステップS804）。

【0177】次に、開発者は、サーバ1の側のPERC

1の開発言語の型のものに変換すれば、引数が1個から3個のいずれの個数のメソッドも呼び出すことができる。

【0170】[id(8)] VARIANT getdata(VARIANT* param0, [optional] VARIANT* param1, [optional] VARIANT* param2);

【0171】10) 仮想関数

この実施例において、仮想関数を実現するための仮想関数の例を以下に示す。ここで、“CPerson”というクラスが、“CEmployee”の親クラスとなる。また、以下のプログラム中の“d_Ref<Person>”は、スキーマクラス<Person>のオブジェクトを指すポインタを保持する変数の型である。

【0172】

IOの環境を用いて、或いはオートメーションサーバ14を介してクライアント2の側から操作することによって、データベース11に実体（インスタンス）を格納する（ステップS805）。なお、いずれの方法による場合でも、データベース11に格納したインスタンスは、Visual Basicなどのスクリプト的なインタプリタ言語の実行によって、データベース11のデータやメソッドなどの振る舞いを確認することができる（ステップS806）。

【0178】次に、開発者は、PERCIOのAPIを直接用いて、上記のプログラム自動生成装置3が生成するオートメーションサーバ14と異なる独自のオートメーションサーバを生成して、両方のオートメーションサーバを併用するデータベースアプリケーションを構築する。

【0179】次に、開発者は、クライアント2をコントロールするための、“ActiveX”コントロールを実装する（ステップS808）。なお、このシステムはオートメーション通信の機能を用いた環境として構築されているため、クライアント2の側で動作させる“ActiveX”コントロールをアプリケーションの部品として用いることができる。また、アプリケーションに有用な既存の“ActiveX”コントロールが存在すれば、その再利用も可能である。

【0180】次に、上記のステップで開発した、プログ

ラム自動生成装置 3 を用いて作成したオートメーションサーバ 1 4、P E R C I O の A P I を用いて生成したオートメーションサーバ、及び“ActiveX”コントロールをクライアントプログラム 2 1 の中で統合させる。ここでは、さらに V i s u a l B a s i c などのスクリプト的なインタプリタ言語を用い、アプリケーションのテスト実行、G U I のプロトタイプ作成などを行う（ステップ S 8 0 9）。

【0 1 8 1】ステップ S 8 0 9 までで、この実施例のシステムのためのデータベースアプリケーションは一応完成するが、オートメーション通信のためのプログラムの部分は、V i s u a l C ++ などのコンパイラ言語でも開発可能なので、開発者は、このようなコンパイラ言語を用いて、テストやプロトタイプでない最終的なデータベースアプリケーションを構築する（ステップ S 8 1 0）。

【0 1 8 2】

【発明の効果】以上説明したように、本発明によれば、オブジェクト指向データベースを操作するためのプログラムを容易に作成することができる。また、そのプログラムの保守も容易になる。また、外部操作情報の生成によって、クライアントサーバ型のシステム構成をした場合のクライアント側のプログラムの作成も容易になる。

【0 1 8 3】また、本発明は、オートメーション通信の機能を利用することによって、既存のシステムの利用が可能となり、プログラムの開発、運用及び保守が容易になる。

【0 1 8 4】さらに、本発明は、インターフェース仕様を統一することによって、データベースに定義されているスキーマを構成するオブジェクトが多相性をサポートしている場合に、このオブジェクトの多相性の機能を十分に利用することができる。

【図面の簡単な説明】

【図 1】本発明の実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

【図 2】図 1 に示すシステムを構築する手順を示すフロー

ーチャートである。

【図 3】図 1 のプログラム自動生成装置によるプログラムの生成の処理の概要を示すフローチャートである。

【図 4】プロパティの取得関数の生成手順を示すフローチャートである。

【図 5】プロパティの設定関数の生成手順を示すフローチャートである。

【図 6】オートメーションオブジェクトのメソッドの生成手順を示すフローチャートである。

10 【図 7】オートメーションオブジェクトの O D L 記述ファイルの生成手順を示すフローチャートである。

【図 8】図 1 のクライアントプログラムからデータベースへアクセスするための処理の流れを説明する図である。

【図 9】図 1 のオートメーションサーバによるメソッドの呼び出し手順を示す図である。

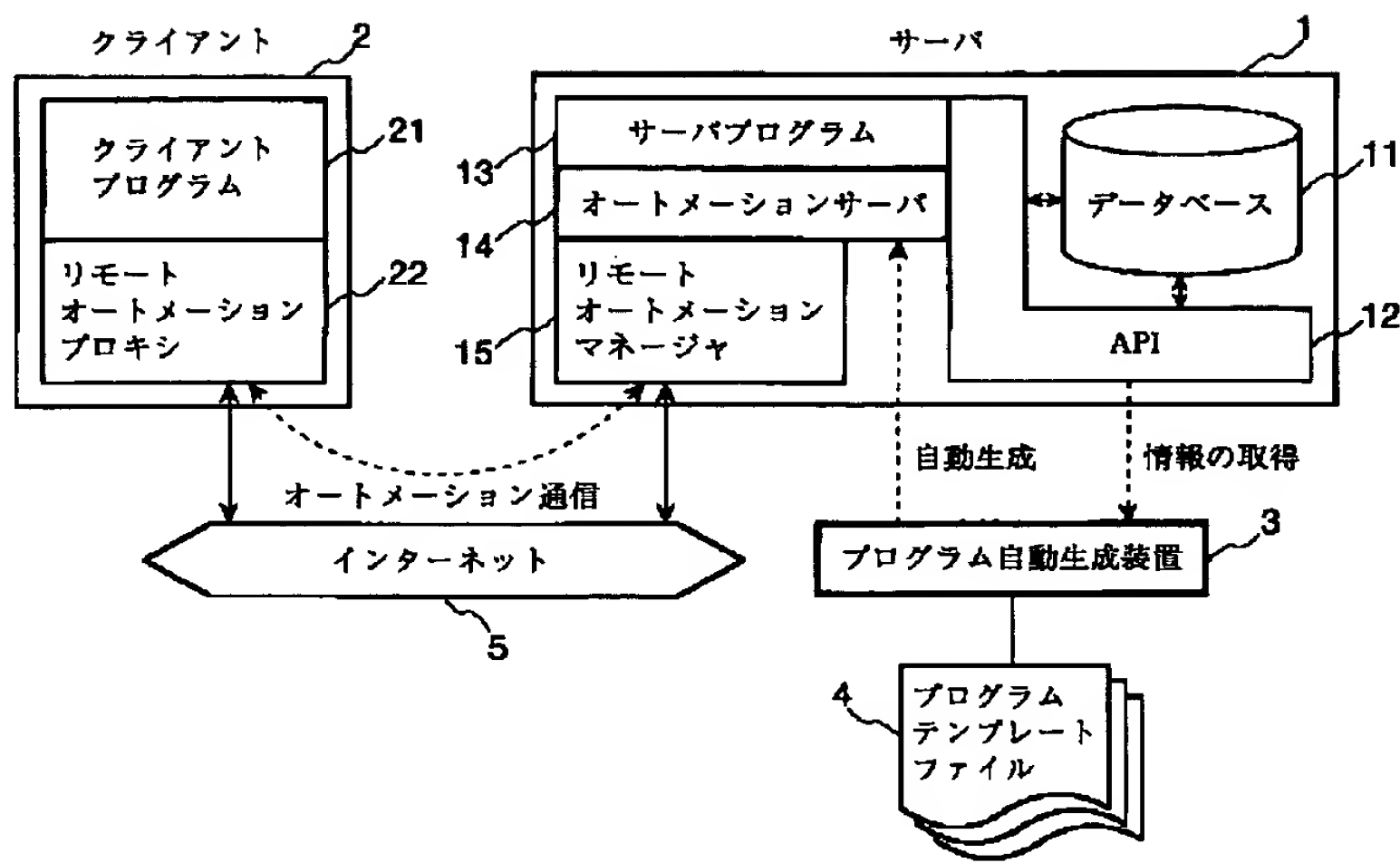
【図 1 0】本発明の実施の形態の具体例にかかるクライアントサーバ型のデータベース操作システムの開発プロセスを示すフローチャートである。

20 【図 1 1】本発明の他の実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

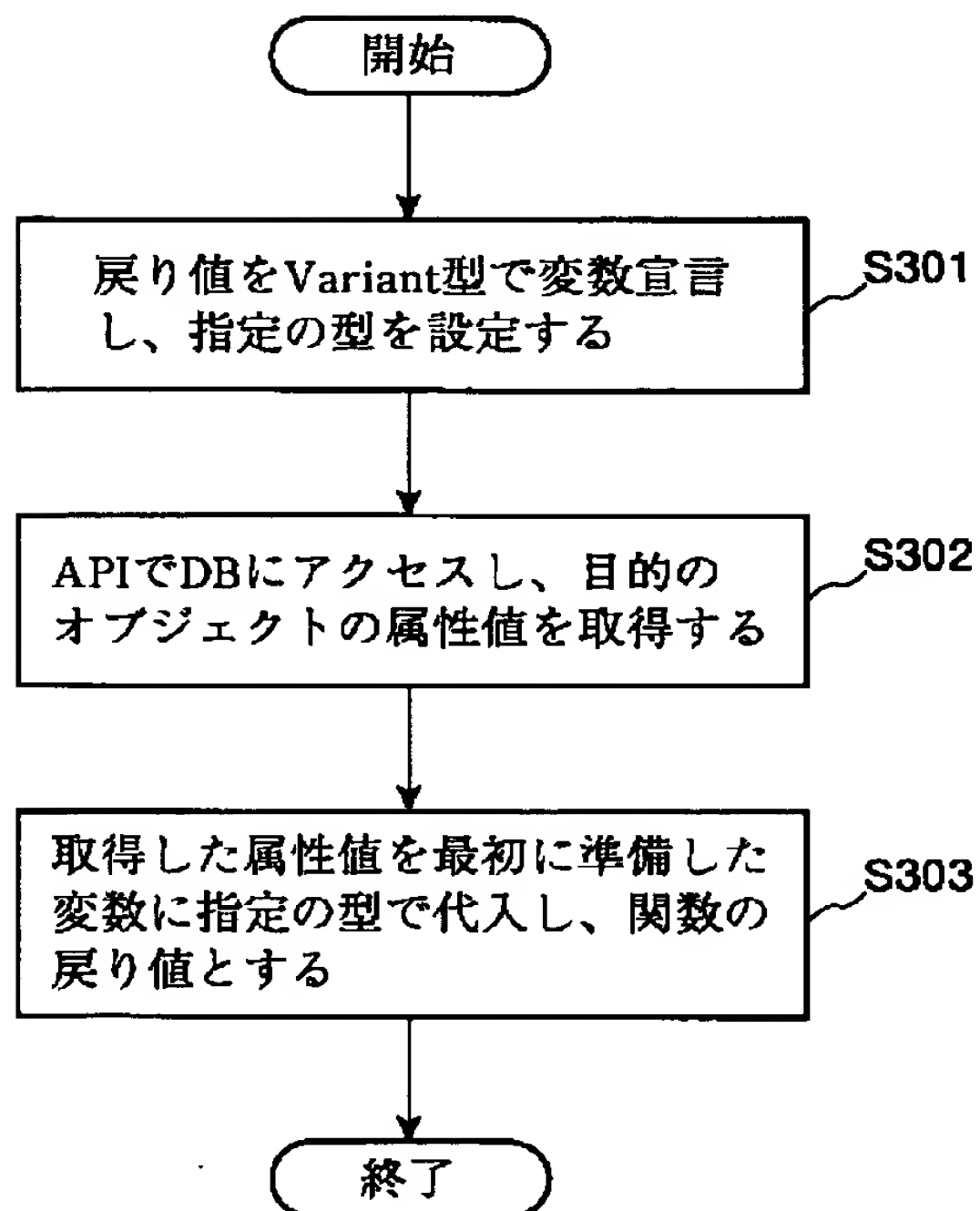
【符号の説明】

- | | |
|--------|--|
| 1 | サーバ |
| 2 | クライアント |
| 3 | プログラム自動生成装置 |
| 4 | プログラムテンプレートファイル |
| 5 | インターネット |
| 6 | スタンドアローン型のコンピュータシステム |
| 30 1 1 | データベース |
| 1 2 | A P I (Application Programing Interface) |
| 1 3 | サーバプログラム |
| 1 4 | オートメーションサーバ |
| 1 5 | リモートオートメーションマネージャ |
| 2 1 | クライアントプログラム |
| 2 2 | リモートオートメーションプロキシ |

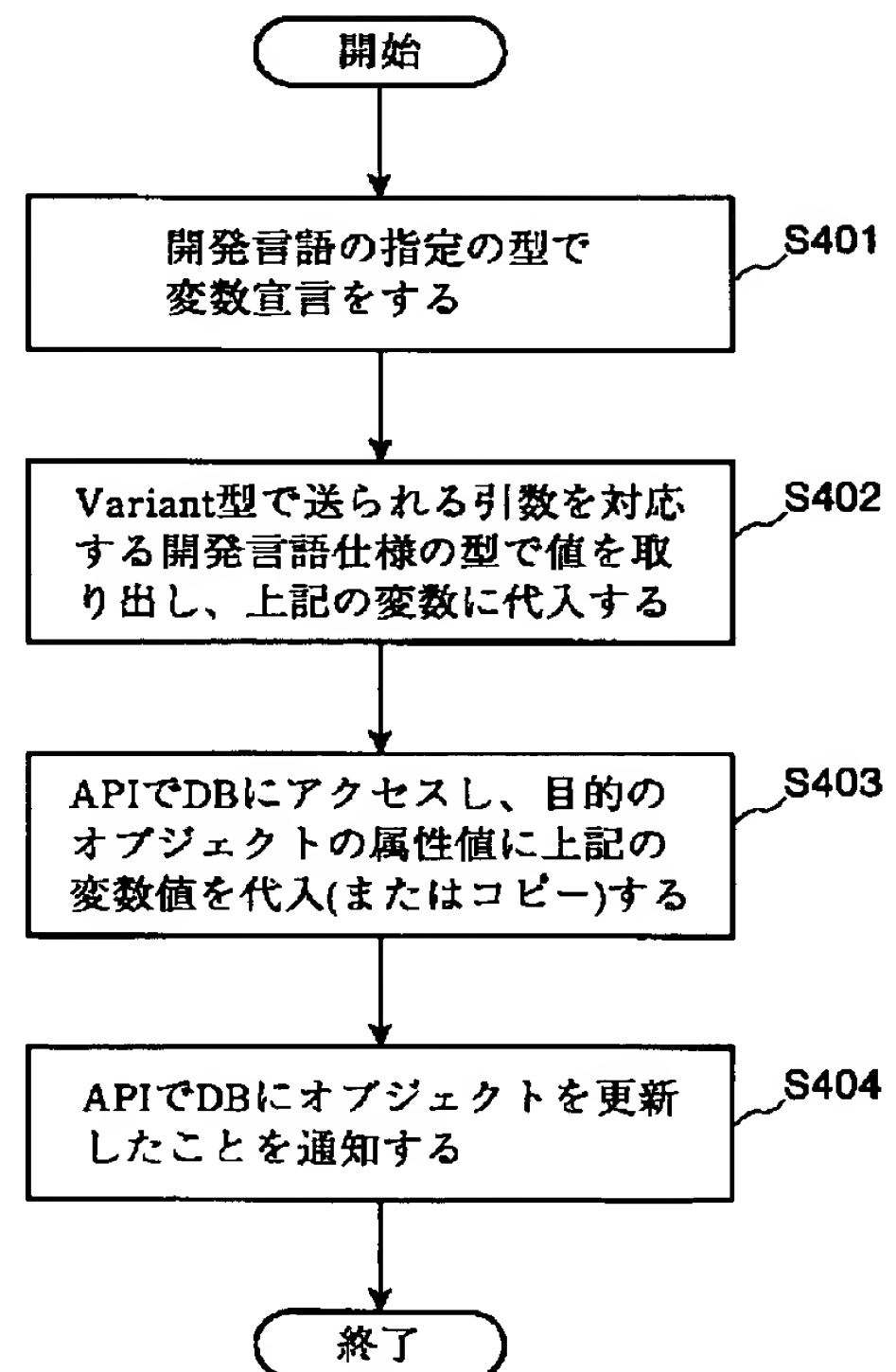
【図 1】



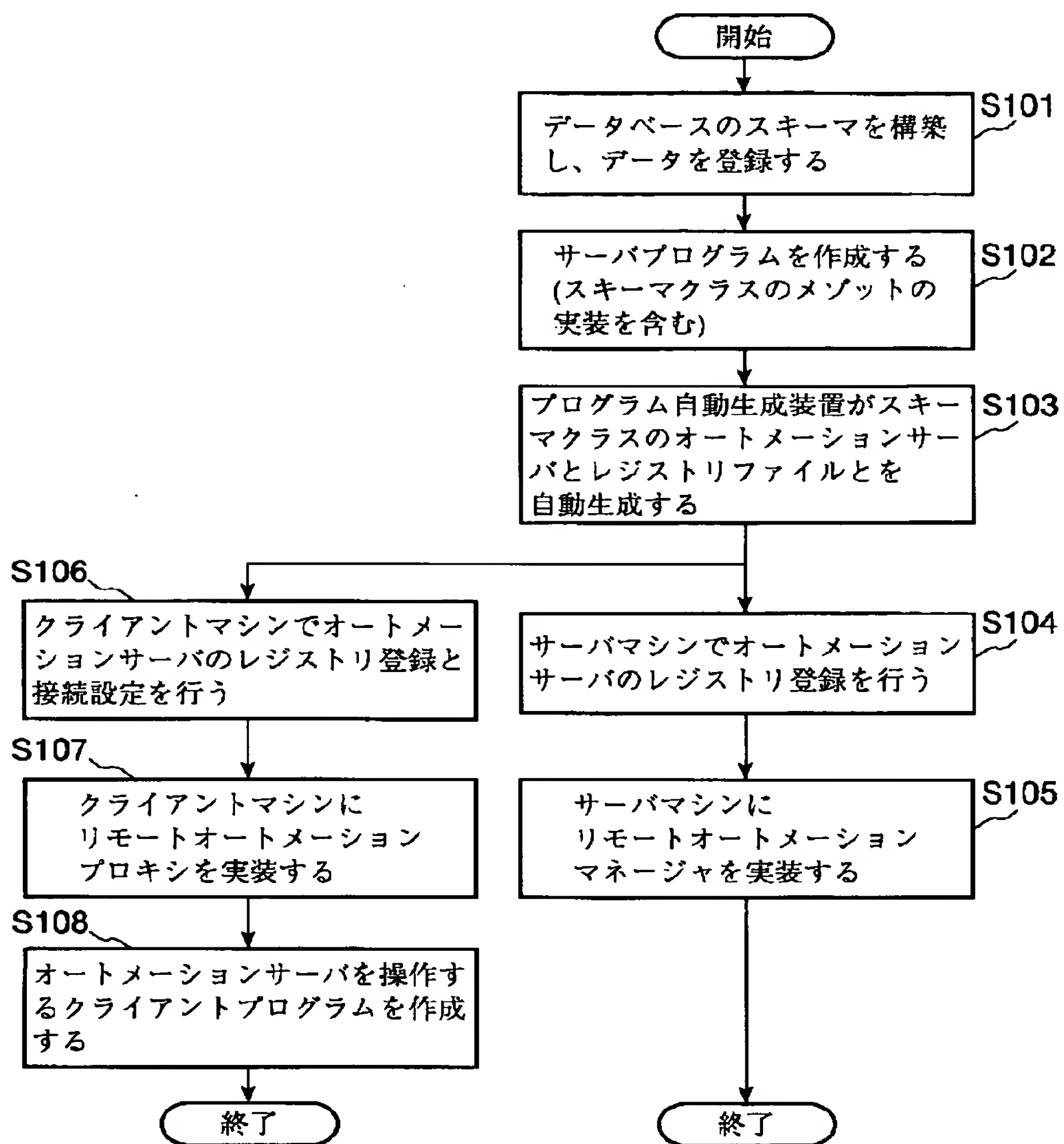
【図 4】



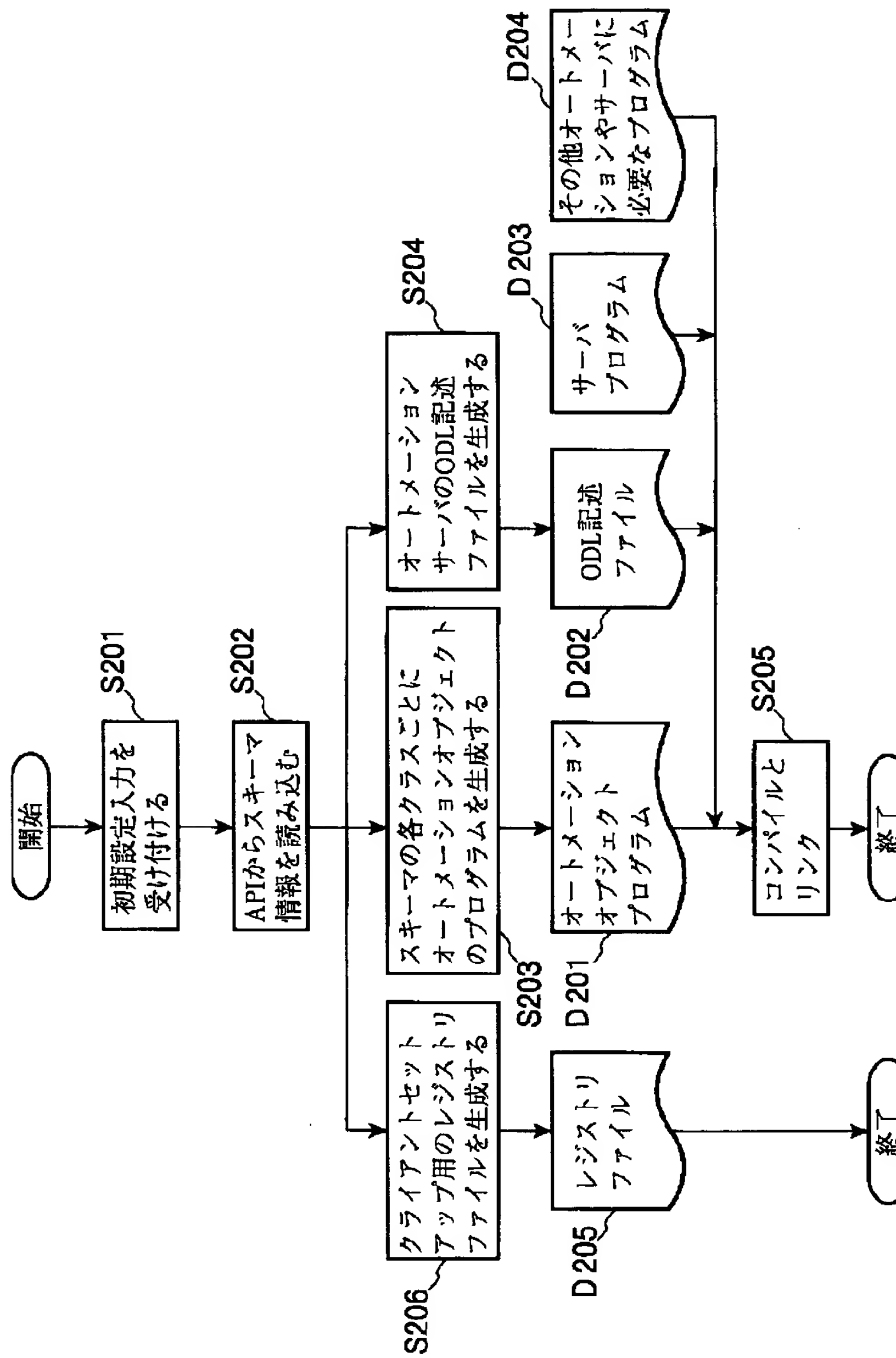
【図 5】



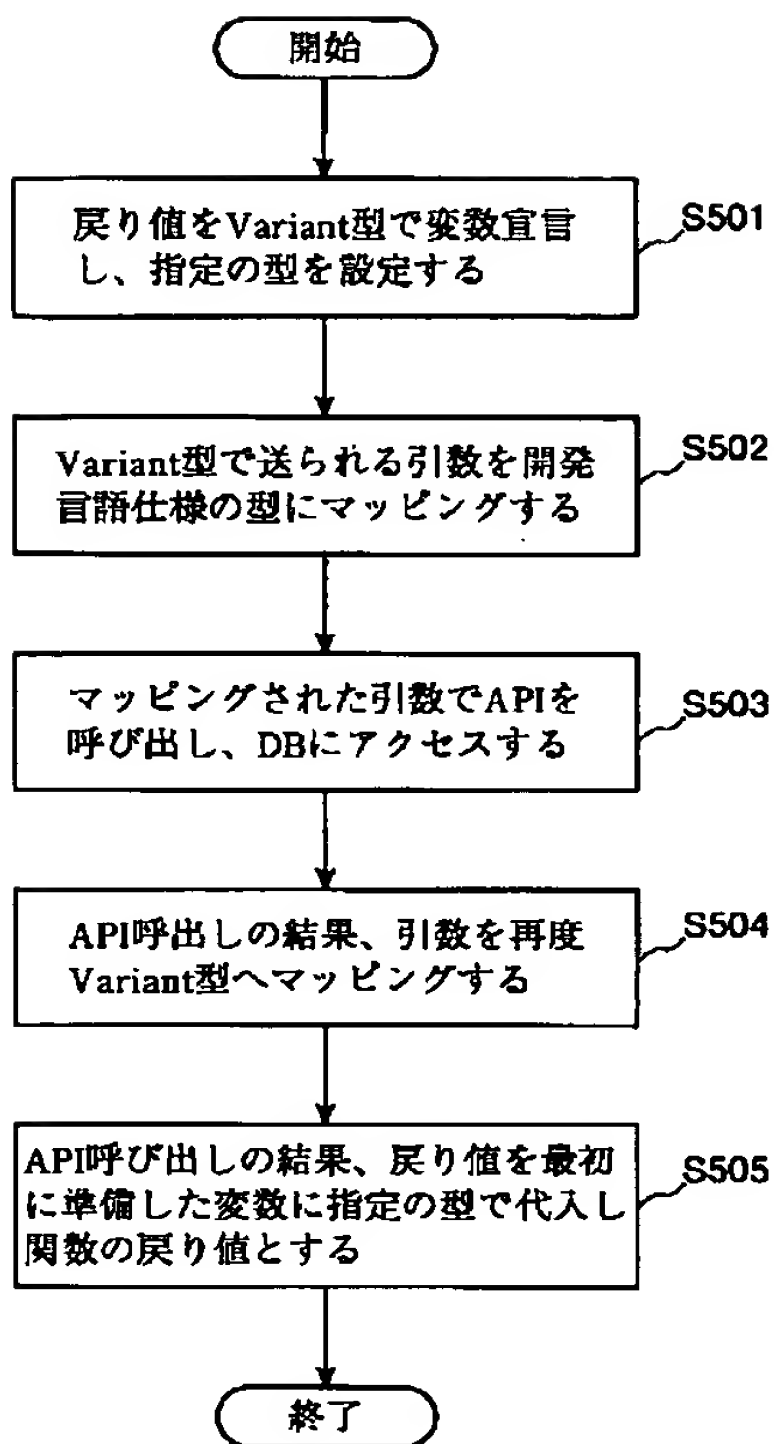
【図 2】



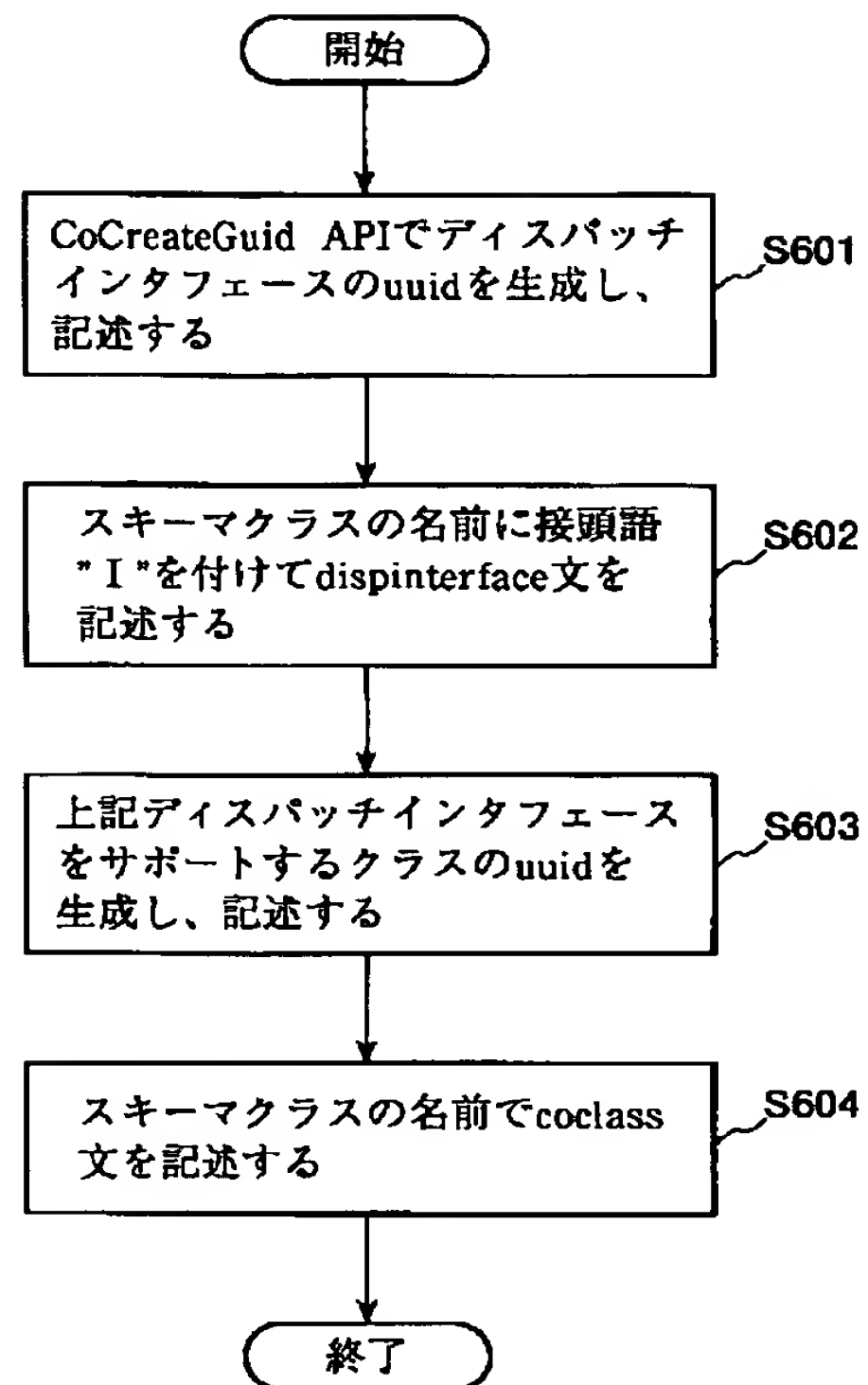
【図 3】



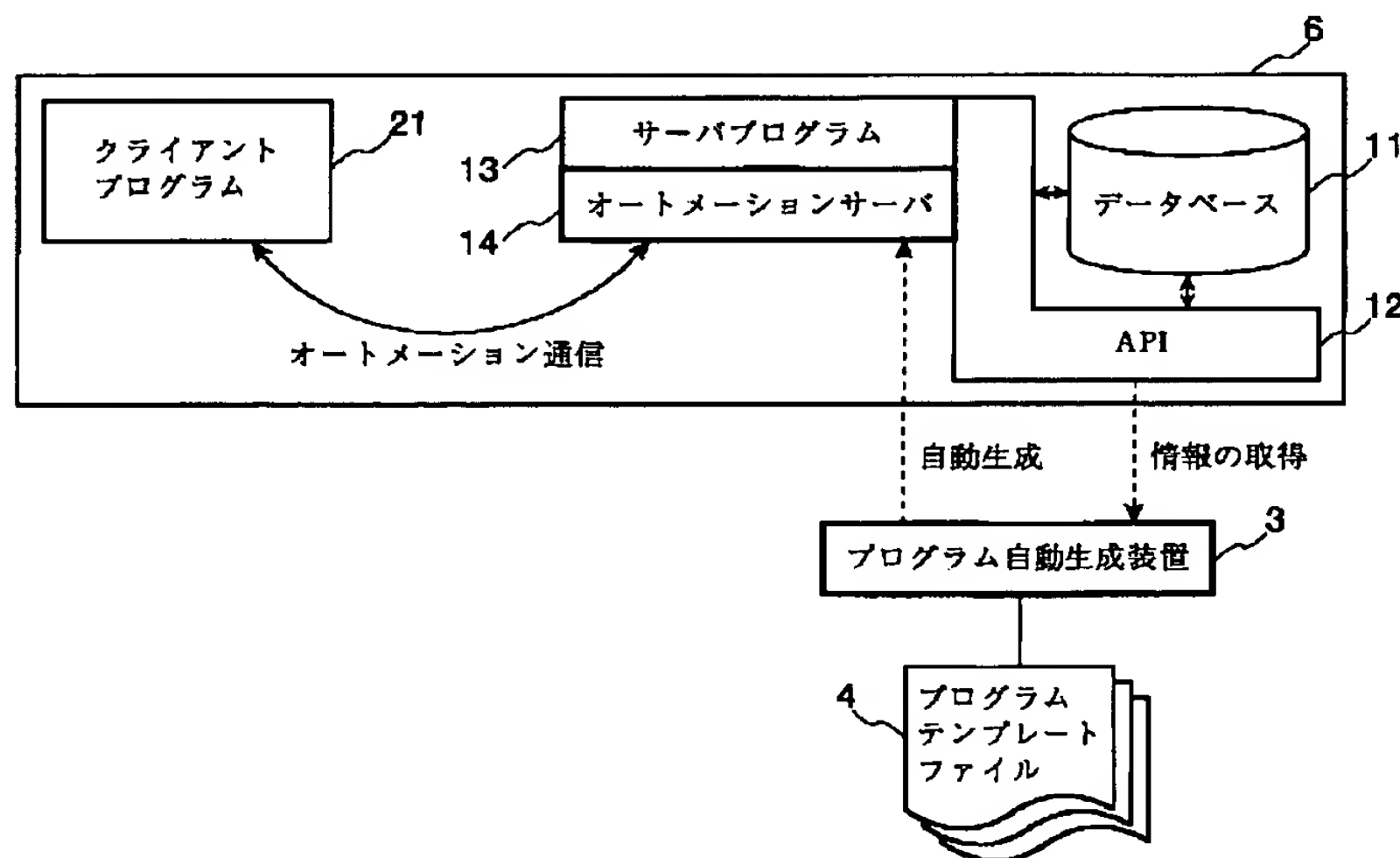
【図 6】



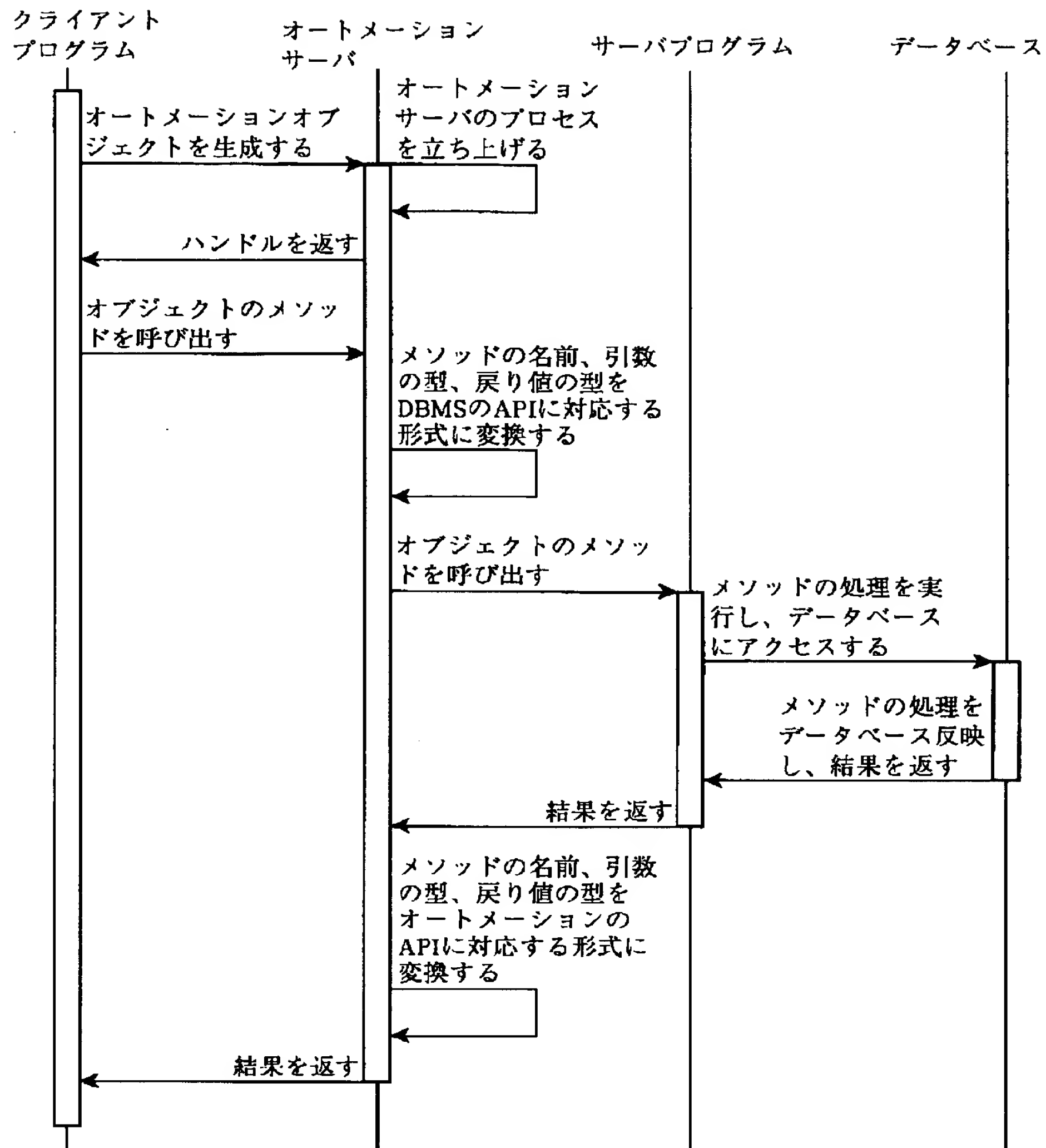
【図 7】



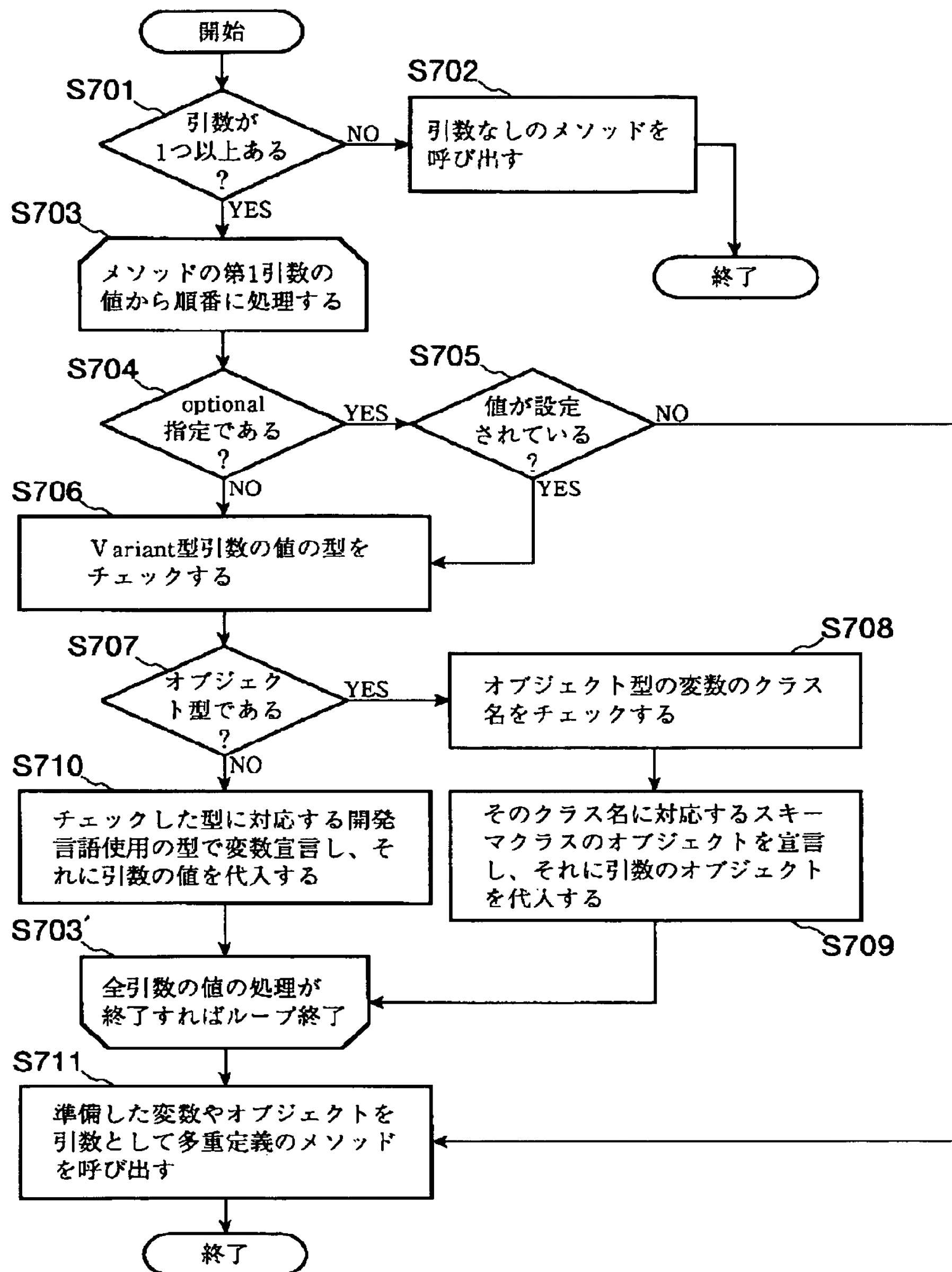
【図 1 1】



【図 8】



【図 9】



【図 1 0】

